THE BENCHMARKED LINEARIZED EQUATIONS OF MOTION FOR AN IDEALIZED BICYCLE (IMPLEMENTED IN SOFTWARE AND DISTRIBUTED VIA THE INTERNET)

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Andrew Erwin Dressel

January 2007

© 2007 Andrew Erwin Dressel

ABSTRACT

People have been successfully building and riding bicycles since the 1800s, and many attempts have been made to describe the motion of these machines mathematically. However, common acceptance of the correct linearized equations of motion for a bicycle has remained elusive.

In his 1988 master's thesis at Cornell University, Scott Hand derived the equations again and performed the first known extensive survey of the literature, finding and documenting the mistakes made in previous attempts. The question remained however of what mistakes, if any, Mr. Hand and his advisors made.

The subsequent advent of cheap and plentiful computing power and the development of numerical methods to take advantage of it provide an opportunity to confirm, once and for all, the correct linearized equations of motion for an idealized bicycle. That is exactly what A. L. Schwab, J. P. Meijaard, and J. M. Papadopoulos have done in their recent paper.

The next step is to efficiently promulgate these correct and confirmed equations in a useful form. The goal is that anyone working in the areas of bicycle or motorcycle handling or control can use these equations directly or verify their own underlying equations against this benchmark.

This thesis describes a program, JBike6, its on-line help, and its web site designed specifically for that purpose: to provide a turn-key application for evaluating the self-stability of a bicycle. JBike6 also generates numbers (eigenvalues and matrix entries) that can be used to compare, to very high precision, against any other linearized or fully non-linear equations of motion for a bicycle.

After a brief review of the application, theory, and results of JBike6, the contents of this thesis consist primarily of hard copy of the on-line help and web site and screen shots of the program. The text has been modified to be more readable as a

narrative and some pictures have been formatted to fit within the margins. Obviously, the interactive nature of the program, the help file, and the web site, including the hyperlinks, animations, and videos, is not available in this printed document.

While all the components will continue to evolve, this thesis is a snapshot of them in September 2006. Many redundancies have been removed, but some remain in order to preserve the integrity and flow of the individual components.

All these components may currently be found on-line at www.tam.cornell.edu/~ad29/JBike6

BIOGRAPHICAL SKETCH

Andrew E. Dressel was born on July 26, 1962, in Meriden, Connecticut. He attended public schools in Cheshire, Connecticut, and he enrolled in Rensselaer Polytechnic Institute in Troy, New York in the fall of 1980. After six semesters at RPI; studying for one semester as an exchange student at Williams College in Williamstown, Massachusetts; and working for three semesters and three summers at different software companies, including IBM in Boca Raton, Florida; he graduated from Rensselaer with a Bachelors Degree in Computer Science in May 1985.

Soon after graduation he was invited to join some previous classmates in starting a company to develop and market an in-car navigation system. This company grew and evolved to become MapInfo Corporation, "the leading provider of location intelligence solutions, integrating software, data and services to provide greater value from location-based information and drive more insightful decisions for businesses and government organizations around the world."

After the fall of the Berlin wall and the disintegration of the Soviet Union, Andrew spent two years (1992-1994) living in Kiev, Ukraine to promote technological entrepreneurship by establishing a business incubator at Kiev Polytechnic Institute.

When he returned from Kiev, Andrew started Software Factory International with a partner, Mark Kapij who had established the L'viv Business Incubator, to promote and support software companies operating in the two Ukrainian Incubators. By 1997, the companies had matured and the Ukrainian communication and banking infrastructure had developed to the point that SFI was no long necessary and Andrew returned to MapInfo as a product manager.

In 2001, after a rewarding career in software development, Andrew left MapInfo to pursue new interests. Because primary among these were bicycles, he chose Cornell University's Human Power and Biomechanics Laboratory (now

Biorobotics and Locomotion Lab) in the Department of Theoretical and Applied Mechanics. After one semester of undergraduate engineering classes, he entered the one-year Master of Engineering program.

He received his M.Eng. degree in May 2003 and decided to stay at Cornell University for another year to learn more about bicycle dynamics. After completing the necessary coursework, he left Ithaca in June of 2004, to work on his thesis at his leisure and learn to fly hot-air balloons. He received his hot-air balloon pilots' license in 2005, and is now Service Manager at the Wheel & Sprocket bike shop in Brookfield, Wisconsin, just outside of Milwaukee, Wisconsin.

ACKNOWLEDGMENTS

I would especially like to thank:

Andy Ruina, Professor of Theoretical and Applied Mechanics at Cornell University, chief of the The Biorobotics and Locomotion Lab, and my advisor.

Arend Schwab, Assistant Professor in Applied Mechanics at of Delft University of Technology and creator of TAM 674: Applied Multibody Dynamics, the class in which we derived the non-linear bicycle model and implemented it in MATLAB. Also co-author of JBike6.

Jim Papadopoulos, author of JBike5, co-author of JBike6, tireless proof reader, and incredible font of bicycle knowledge.

Professor Francis C. Moon, thesis committee member, for his excellent suggestions of questions to answer and additional material to include.

The rest of the Theoretical and Applied Mechanics Biorobotics and Locomotion Lab at Cornell University; Mario Gomes, Dave Cabrera, Manoj Srinivasan, and Sam Walcott; for their patience.

Geoff Recktenwald , fellow TAM graduate student, TAM 674 classmate, and MATLAB debugger extraordinaire.

Mike and John Miller for teaching me how to ride hard.

Tom Muro, Mark Kapij, and John Chase for their continuous encouragement throughout my adventure in graduate school.

The MapInfo crew; Laszlo Bardos, John Haller, Mike Marvin, and Sean O'Sullivan; for teaching me how to do the seemingly impossible.

Fratney Miller, my best friend, for not letting me forget to finish.

Erwin and Ellen Dressel, my parents, for being happy with just about whatever I do.

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

JBike6 is a tool, which can be used to calculate and examine the eigenvalues (i.e., perturbation-growth exponents) for an idealized, uncontrolled bike. It has been developed in MATLAB, a computer language and development environment for computationally intensive tasks from The MathWorks.

It uses physical parameters (pre-defined or user-entered) to describe a particular bike, then calculates and plots the linearized perturbation-growth eigenvalues over a specified range of forward speeds.

An idealized, rigid, uncontrolled bike with rigid rider has four eigenvalues and, depending on the forward speed, they are either: all real (non-oscillatory); two real plus a complex pair representing oscillatory motion; or in rare cases two complex pairs. (Idealized means that the bodies are perfectly rigid and symmetrical about the midplane, the joints are frictionless, and it rolls on knife-edge wheels without loss due to friction and without slipping on a smooth, rigid, horizontal plane.)

In general, a bike design is self-stable at those forward speeds for which the real parts of all the eigenvalues are less than zero. (Self-stable means that without external input, it will eventually roll straight and upright even if perturbed.) Such behavior can easily be seen in many real-world bicycles, when rolled without a rider.

In addition to determining the self-stability of a particular bicycle design, the eigenvalues or matrix entries that JBike6 generates can be used to confirm the validity of any other bicycle model that can produce eigenvalues or similar matrix entries. An example of how to accomplish this is given in an appendix on page 98.

CHAPTER TWO

THEORY

A bike is a non-holonomic system because of how its wheels interact with the ground: its configuration is path-dependent. In order to know its exact configuration, especially location, it is necessary to know not only the configuration of its parts, but also their histories: how they moved over time. This complicates mathematical analysis.

A bike is also an example of an inverted pendulum. Thus, just as a broomstick is easier to balance than a pencil, tall bikes can be easier to balance than short ones because their characteristic time of falling is longer.

The idealized model of a bike used by JBike6 is a conservative system: there is no energy dissipation. At forward speeds for which weave oscillations decrease with time, the energy is being converted from the weave into an increase in the forward speed. Although it is conservative, it is not Hamiltonian due to the non-holonomic constraints mentioned above.

Finally, in the language of control theory, a bike exhibits non-minimum phase behavior. To turn to the right, you must initially turn to the left and visa versa.

Model

The idealized rigid-body dynamics model of the bike is defined as four interconnected rigid bodies interacting with a horizontal ground plane:

1. A rear wheel with left-right mass symmetry and polar mass symmetry, rolling on a knife edge, and revolutely connected to the rear frame.

2. A rear-frame, rider, rear rack, all in one rigid assembly, with left-right mass symmetry, revolutely connected to the front fork. (Although JBike6 uses only the

entire assembly's mass and inertia, a frame, rider, and rack may be specified in case data exists only for the separate sub-components.)

3. A front-fork, handlebar, and front basket, all in one rigid assembly, with left-right mass symmetry, revolutely connected to the front wheel.

4. A front wheel with left-right mass symmetry and polar mass symmetry, rolling on a knife edge.

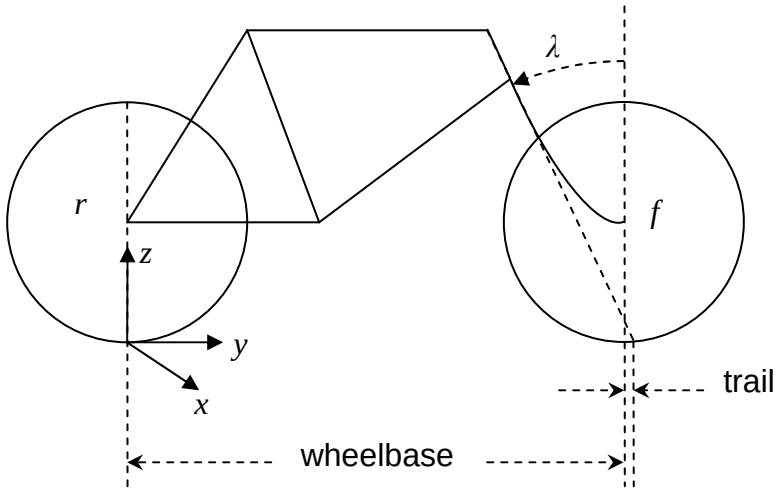
In the complete bike assembly, when held upright and steered straight, all components share the same symmetry midplane. The wheel axles are normal to that plane and the steer axis is in that plane.

All the revolute joints are frictionless, and the rolling occurs without dissipation or slippage. There is no propulsive or braking torque, and no slope or air resistance.

JBike6 uses a linearized analysis for small deviations from the upright straight condition. High accuracy can be expected only when steer and lean angles are infinitesimal, however many results and trends [easy to find because the governing equations are linear] also hold quite well for realistically large angles. Determining the inaccuracy resulting from realistic angles is the province of nonlinear analysis.

Note that in a linearized analysis near the upright straight condition, the bike's full geometry and mass distribution is not needed – for example, there is no 'pitch' motion of the rear frame assembly, so that component of inertia is irrelevant. Also, apart from establishing the rotation rate of the wheels, wheel diameter is unimportant – one could as easily envision 'infinitesimal' wheels placed at the bike's two contact points. JBike6 takes full data as input so that they can be used in comparison with other systems without confusion about what parameters are used for what.

The two degrees of freedom are q=[lean angle, steer angle], with lean to the right (clock-wise when viewed from the rear) and steer to the left (counter-clock-wise when viewed from above) as positive when driving forward.



Derivation of Equations

The linearized equations of motion for an idealized bicycle implemented in JBike6, as derived by Scott Hand in his thesis, can be expressed as a system of two, coupled, ordinary second-order differential equations with constant coefficients. Hand's derivation uses Lagranges's equations with Lagrange multipliers and "follows the major steps taken in the derivation given by Neĭmark and Fufaev [1967]"[1], except that it corrects the potential energy expression and other mistakes. He uses constrained Lagrange equations because the four interconnected bodies "complicates the Newtonian analysis significantly", and then eliminates the constraints. As far as he knew, his was the "first set of equations derived using Langrangian equations which agrees in full with other Newtonian derivations."

[1] Hand [1988]

Some of the expressions for kinetic and potential energy cannot be solved in closed form so small-angle approximations are used. Also, any terms which "do not contribute to Langrange's equations" are omitted along the way. In fact, due to the complexity of the kinematics, there is no known example of systematic linearization of fully non-linear equations of motion of a bicycle.

Although specifying the location and orientation of all 4 bodies in 3D requires 24 (= 4 * [3 + 3]) coordinates, there are only 2 coordinates required to describe the configuration, independent of location and direction, of the bike at any time:

1. lean angle of the rear frame relative to ground plane,
2. steer angle of the front fork relative to the rear frame, and

The equations can be represented as:[2]

$$M_{\theta\theta}\ddot{\theta}_r + K_{\theta\theta}\theta_r + M_{\theta\psi}\ddot{\psi} + C_{\theta\psi}\dot{\psi} + K_{\theta\psi}\psi = M_{\theta_r} \quad \text{(the 'lean' equation)}$$

$$M_{\psi\psi}\ddot{\psi} + C_{\psi\psi}\dot{\psi} + K_{\psi\psi}\psi + M_{\psi\theta}\ddot{\theta}_r + C_{\psi\theta}\dot{\theta}_r + K_{\psi\theta}\theta_r = M_{\psi} \quad \text{(the 'steer' equation)}$$

where

θ_r is the lean angle of the rear assembly

ψ is the steer angle of the front assembly relative to the rear

M_θ and M_ψ are moments (torques) applied at the rear assembly and the steering axis, respectively. For the analysis JBike6 performs of an uncontrolled bike, these are both taken to be zero.

Note that for the upright bike $C_{\theta\theta} = 0$, and so it does not appear in the second-order ordinary differential equations above. The rest of the coefficients are defined as follows:

$$M_{\theta\theta} = T_{yy} \qquad M_{\theta\psi} = F'_{\lambda y} + \frac{c_f}{c_w} T_{yz}$$

$$M_{\psi\theta} = F'_{\lambda y} + \frac{c_f}{c_w} T_{yz} \qquad M_{\psi\psi} = F'_{\lambda\lambda} + 2\frac{c_f}{c_w} F''_{\lambda z} + \frac{c_f^2}{c_w^2} T_{zz}$$

[2] Hand [1988]

$$C_{\theta\theta} = 0 \qquad C_{\theta\psi} = -C_{\psi\theta} + V\left(T_{yz}\frac{\cos\lambda}{c_w} - \frac{c_f}{c_w}m_t\bar{h}_t\right)$$

$$C_{\psi\theta} = V\left(\frac{C_f}{a_f}\cos\lambda + \frac{c_f}{c_w}\left(\frac{C_f}{a_f} + \frac{C_r}{a_r}\right)\right) \qquad C_{\psi\psi} = V\left(\frac{\cos\lambda}{c_w}\left(F''_{\lambda z} + \frac{c_f}{c_w}T_{zz}\right) + \frac{c_f}{c_w}\upsilon\right)$$

$$K_{\theta\theta} = -gm_t\bar{h}_t \qquad K_{\theta\psi} = g\upsilon - V^2\left(H_t\frac{\cos\lambda}{c_w}\left(\frac{C_f}{a_f} + \frac{C_r}{a_r}\right) - \frac{\cos\lambda}{c_w}m_t\bar{h}_t\right)$$

$$K_{\psi\theta} = g\upsilon \qquad K_{\psi\psi} = -g\upsilon\sin\lambda + V^2\left(\frac{C_f}{a_f}\sin\lambda\frac{\cos\lambda}{c_w} + \upsilon\frac{\cos\lambda}{c_w}\right),$$

$\upsilon = m_f d + \dfrac{c_f m_t \bar{\ell}_t}{c_w}$ and $d = \bar{h}_f \sin\lambda + \bar{\ell}_f \cos\lambda - c_f$ for simplification

V is the forward speed, and g is the gravitational constant.

c_w is the wheelbase and c_f is the trail: the distance from where steering axis intersects ground to the front wheel contact point.

λ is steering axis tilt measured back from vertical (90° - head angle).

T_{yy}, T_{yz}, T_{zz} are inertia tensor components for entire bike about the rear wheel contact point: $T_{yy} = m_r\bar{h}_r^2 + R_{yy} + m_f\bar{h}_f^2 + F_{yy}$.

R_{yy} and F_{yy} are inertia tensor components of the rear and front respectively.

C_r and C_f are polar mass moment of inertia of rear and front wheels respectively.

$F'_{\lambda\lambda}$ is the mass moment of inertia about the steering axis of the front assembly.

$F'_{\lambda y}$ is the dot product of front assembly inertia tensor with unit vectors in Y and λ directions: i. e. the torque about steering axis for lean acceleration.

$F''_{\lambda y}$ is the dot product of front assembly inertia tensor with unit vectors in Z and λ directions: i. e. torque about steering axis for yaw acceleration.

m_r, m_f, and m_t are the mass of the rear assembly, front assembly, and total mass, respectively: $m_t = m_r + m_f$.

$\bar{h}_r$, $\bar{\ell}_r$, $\bar{h}_f$, $\bar{\ell}_f$, $\bar{h}_t$, and $\bar{\ell}_t$ are the height and forward distance to the center of mass of the rear assembly, front assembly, and total mass, respectively: $\bar{\ell}_t = \dfrac{m_r \bar{\ell}_r + m_f \left(c_w + \bar{\ell}_f\right)}{m_t}$.

a_f and a_r are the radii of the front and rear wheels, respectively.

Note that some of the *K* and *C* coefficients have been slightly rearranged from Hand's thesis to show better the symmetries and velocity dependences. See the appendix on page 23 below for details.

These equations of motion can also be written in matrix form:

$$\left[\overbrace{\begin{pmatrix} M_{\theta\theta} & M_{\theta\psi} \\ M_{\psi\theta} & M_{\psi\psi} \end{pmatrix}}^{M} D^2 + \overbrace{\begin{pmatrix} C_{\theta\theta} & C_{\theta\psi} \\ C_{\psi\theta} & C_{\psi\psi} \end{pmatrix}}^{C} D + \overbrace{\begin{pmatrix} K_{\theta\theta} & K_{\theta\psi} \\ K_{\psi\theta} & K_{\psi\psi} \end{pmatrix}}^{K} \right] \begin{pmatrix} \theta_r \\ \psi \end{pmatrix} = \begin{pmatrix} M_{\theta_r} \\ M_{\psi} \end{pmatrix}$$

and more-compactly as $M\ddot{q} + C\dot{q} + Kq = f$

Where

M is the symmetrical mass matrix which contains terms that include only the mass and geometry of the bike,

C is the so-called damping matrix, even though the system has no dissipation, which contains terms that include the forward speed *V* and is asymmetric,

K is the so-called stiffness matrix which contains terms that include the gravitational constant *g* and V^2 and is symmetric in *g* and asymmetric in V^2,

D is the differential operator,

q is a vector of lean angle and steer angle, and

f is a vector of input torques.

If we assume a solution to the differential equation to have the form $q = q_0 e^{\lambda t}$ in the homogeneous case, then for *q* to be non-trivial, the determinant of the characteristic equation must be zero: $\det(M\lambda^2 + C\lambda + K) = 0$

Then the solutions to this equation, λ, are the four eigenvalues. To find these solutions, JBike6 recasts the 2 second-order equations as 4 first-order equations, combining the *M*, *C*, and *K* matrices into 2 4x4 matrices, *A* and *B*:

$$B = \begin{bmatrix} M_{\theta\theta} & M_{\theta\psi} & 0 & 0 \\ M_{\psi\theta} & M_{\psi\psi} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } A = \begin{bmatrix} -C_{\theta\theta} & -C_{\theta\psi} & -K_{\theta\theta} & -K_{\theta\psi} \\ -C_{\psi\theta} & -C_{\psi\psi} & -K_{\psi\theta} & -K_{\psi\psi} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

It then calculates the 4 *generalized eigenvalues* of them via: $Ax = \lambda Bx$

Where the values of λ that satisfy the equation are the generalized eigenvalues, and the corresponding values of *x* are the generalized right eigenvectors whose entries can be interpreted as initial conditions: $[\dot{\theta}_r, \dot{\psi}, \theta_r, \psi]^T$.

Note that the eigenvalues are intrinsic to the particular bike, as specified by its parameters, and are completely independent, short of errors or inaccuracies, of the equations used, the methods used to evaluate them, or the coordinate systems and units used to measure the parameters.

For a complete derivation of these equations and a full definition of all the terms, see Hand's thesis, available on-line at http://ruina.tam.cornell.edu/research/topics/bicycle_mechanics/papers/comparisons_stability_analysis.pdf.

Eigenvalues, Eigenvectors, and Eigenmodes

The two, coupled, ordinary second-order differential equations contain non-zero coefficients for the first time derivative (rates of change) of the lean and steer angles. These are in part due to gyroscopic effects. Therefore, the characteristic

equation is quartic and has four eigenvalues. These are distinct except at points (forward speeds) where two may briefly merge to form double roots.

Sample eigenvalues and eigenvectors of a representative bike:

For forward speed v = 0.0 m/s (bike falls over)

Mode names:	Castor	Capsize	Pendulum	Pendulum	
Eigen-values	-5.9289	-2.7812	2.7812	5.9289	
Eigen-vectors	0.0230	1.0000	1.0000	0.0230	lean rates
	1.0000	0.4702	0.4702	1.0000	steer rates
	-0.0039	-0.3596	0.3596	0.0039	lean angles
	-0.1687	-0.1691	0.1691	0.1687	steer angles

For forward speed v = 3.0 m/s (weave grows)

Modes	Castor	Capsize	Weave	Weave	
Values	-12.1980	-2.6646	2.0025 + 1.7814i	2.0025 - 1.7814i	
Eigen-vectors	-0.0104	-1.0000	**0.5635 + 0.2707i**	0.5635 - 0.2707i	lean rates
	1.0000	0.1563	**-0.6090 + 0.3910i**	-0.6090 - 0.3910i	steer rates
	0.0009	0.3753	**0.2242 - 0.0643i**	0.2242 + 0.0643i	lean angles
	-0.0820	-0.0587	**-0.0728 + 0.2600i**	-0.0728 - 0.2600i	steer angles

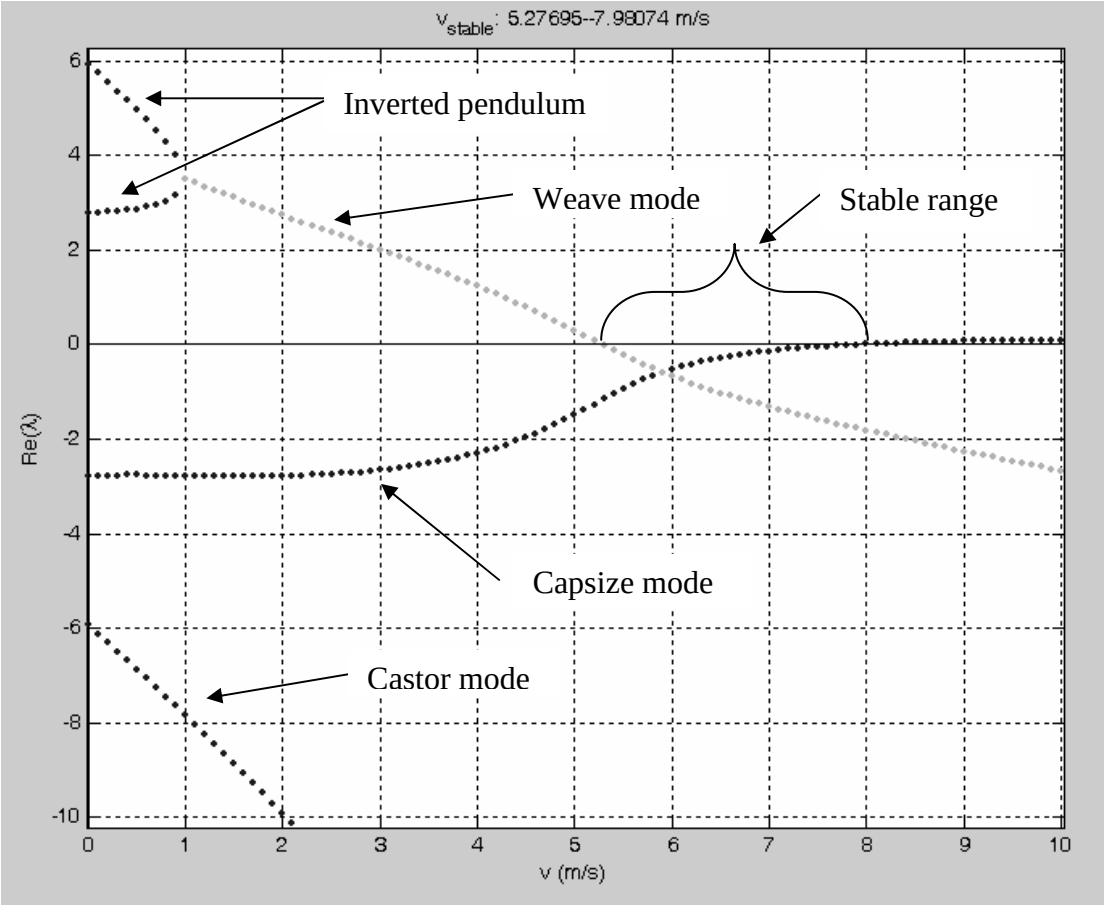
For forward speed v = 6.0 m/s (self-stable)

Modes	Castor	Capsize	Weave	Weave	
Values	-19.8363	-0.5263	-0.6763 + 3.2964i	-0.6763 - 3.2964i	
Eigen-vectors	0.0038	**-0.5263**	**-0.9945 - 0.0055i**	-0.9945 + 0.0055i	lean rates
	-1.0000	**0.1540**	**0.7228 + 0.0049i**	0.7228 - 0.0049i	steer rates
	-0.0002	**1.0000**	**0.0578 + 0.2898i**	0.0578 - 0.2898i	lean angles
	0.0504	**-0.2926**	**-0.0417 - 0.2107i**	-0.0417 + 0.2107i	steer angles

For forward speed v = 9.0 m/s (capsize grows)

Modes	Castor	Capsize	Weave	Weave	
Values	-28.1092	0.0667	-2.2652 + 5.8524i	-2.2652 - 5.8524i	
Eigen-vectors	-0.0009	**0.0667**	-0.9548 - 0.0021i	-0.9548 + 0.0021i	lean rates
	1.0000	-0.0086	0.7428 + 0.2572i	0.7428 - 0.2572i	steer rates
	0.0000	**1.0000**	0.0546 + 0.1420i	0.0546 - 0.1420i	lean angles
	-0.0356	-0.1288	-0.0045 - 0.1252i	-0.0045 + 0.1252i	steer angles

Analysis of the eigenvalues and their corresponding eigenvectors reveals the natural modes of the bike: the eigenmodes. See the table above and the plot below of eigenvalues for a typical utility bike that demonstrates the common characteristics. Dark dots are used to represent strictly real eigenvalues, and light dots represent the real parts of complex conjugate pairs of eigenvalues.



At low forward speeds, starting at zero, the eigenvalues come in two positive-negative pairs and represent the instability of an **inverted pendulum**. Depending on the particular parameters of the bike, lean and steer can have the same or opposite signs, which represent steering away from lean or towards lean, respectively. Also, their rates are positive and so increasing.

At sufficiently higher speed, the two positive real eigenvalues commonly merge to form a complex conjugate pair with positive real parts. This represents unstable oscillatory motion and is referred to as the **weave mode**. The bike leans and steers from side to side, with steering lagging slightly, about 5%, behind leaning as is indicated by the complex eigenvectors, from which the amount of phase lag may be calculated.

As forward speed increases, the frequency of this weave increases, as is indicated by the increasing magnitude of the imaginary parts of the complex conjugate eigenvalues. This increase in magnitude becomes nearly linear with the increase in forward speed, and so the wavelength of the weave is nearly constant. For details on calculating the frequency of the weave and the phase lag between the lean and steer motions, see the appendix on page 24.

For certain bike configurations, at a higher speed still, this pair crosses the real axis and the weave motion becomes stable. This is the beginning of the range of forward speeds for which the bike is self-stable. In the corresponding eigenvector, the lean angle and steer angle have opposite signs, and so, in the model of JBike6, the bike is leaning and steering in the same direction

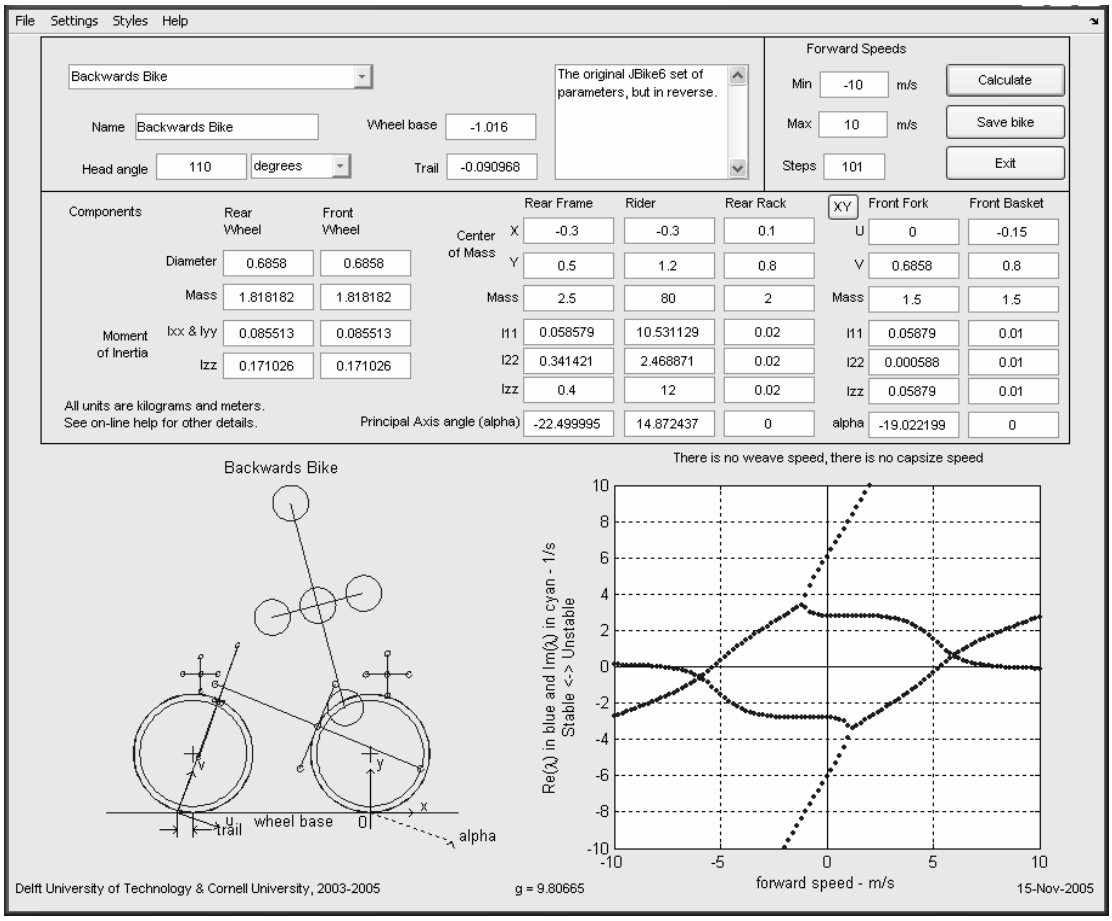
Of the two initially-negative eigenvalues, the lesser one corresponds to the **capsize mode**. For many bike configurations, it becomes positive (unstable) at a speed above the weave speed, marking the end of the self-stable range of speeds. In the corresponding eigenvector, the lean angle and steer angle have opposite signs, and so, in the model of JBike6, the bike is leaning and steering in the same direction. However, while lean rate is positive, steer rate is negative. Also, as the small lean rate value indicates, this instability is very slow, on the order of seconds, and is easily controlled for by a rider.

Finally, the eigenvalue initially most negative has an eigenvector dominated by steer rate and represents the **castor mode**: the tendency of the front wheel to steer in the direction the bike is moving. It only becomes more negative and so more stable as forward speed increases.

CHAPTER THREE

APPLICATION

All of the pre-assembled functionality is available to the user through this single window:



- Select a predefined (previously saved) bike (set of physical parameters).
- Specify a new bike or modify an existing bike and save the parameters.
- Specify the range of forward speeds and the number of speeds to use.
- Calculate and plot the eigenvalues for the specified forward speeds.
- Control the styles used to draw the eigenvalues.

Also, as an uncompiled MATLAB application, JBike6 is distributed via source code that users may modify to alter existing functionality or create new functionality.

Exhaustive detail of all the functionality and examples are provided below in the online help.

Target Audience

The group of people in the world who might benefit from using JBIke6 is pretty small: perhaps a few dozen or so. After being on-line for 6 months, there have been about a dozen downloads of JBike6: mostly academics associated with Papadopoulos, Schwab, or Ruina.

The goal, of course, is that anyone researching the handling or control of two-wheeled, single-track vehicles will use JBike6 either to analyze a particular design, search the design space, or verify their own mathematical model.

It is not clear that anyone in the bicycle industry would use JBike6. First, modern bike design and the rules of thumb that have evolved with it over a hundred years produce quite ridable results. Second, the relationship between the eigenvalues calculated by JBike6 to the handling characteristics as experienced by a rider of a real-world bike is not yet well understood.

Software Architecture

The JBike6 application consists of 3 main parts: the JBike6 program, its on-line help, and the web site to provide its distribution.

The JBike6 program itself consists of two distinct parts:

1. The graphical user interface

   The GUI was created with MATLAB's guide utility and contained in just two modules: JBike6GUI.m and Jbike6GUI.fig. It uses MATLAB's guidedata() function to save parameters in MATLAB's handle structure so that they are accessible to call-back functions. It consists of:

- A main window with input controls and two MATLAB axis controls: one for displaying a stylized representation of the bike based on the input parameters, and another for displaying a plot of the eigenvalues as a function of forward speed.
- Extensive parameter validity checking to make sure that the user cannot enter a value or combination of values that causes the engine to fail.
- A menu system to provide access to additional functionality for which there is not enough room on the main window. For example Print and Plot Styles.
- Additional custom and canned (built into MATLAB) pop-up modal dialogs for user input of additional parameters and styles such as gravitational constant, symbol color, type, and size.
- Tooltips, small windows of text that pop up when the cursor is held over a control, to help explain the controls and parameters.
- Accelerator keys are defined for most menu items so that the user can modify a setting or invoke functionality without having to select a menu item with the mouse. For example: CTRL+G toggles on and off the drawing of grid lines on the eigenvalue plot.
- Functionality to read and write MATLAB script files containing all the parameters necessary to completely describe a particular bike: JBini*.m.
- Functionality to read and write a MATLAB script file containing all the user modifiable settings and plot styles: JBConfig.m.
- Functionality that provides support for multiple versions of MATLAB by checking the version number and calling the appropriate functions, and functionality that provides support for multiple operating systems by

checking for indicators and appropriately launching a browser to view the on-line help.

- Functionality that attempts to catch any unanticipated error that occurs in the engine and handle it as gracefully as possible. This is implemented with MATLAB's try and catch statements.

2. The Engine

   The code to calculate and plot the eigenvalues consists mostly of MATLAB scripts in a handful of modules. As such, it defines most variables in MATLAB's global space and simply sets them and leaves them behind for the next script to access as necessary. This allows a user to run these scripts from the MATLAB command line as desired and access their variables as necessary. The modules include:

   - Jbfig.m draws a representation of the bike in a MATLAB axis on main the Jbike6 main window. If JBike6 is running without the GUI, it draws the bike in a separate window. It uses the following support modules:

     Jbdrawcross.m draws a cross to represent wheel hubs.

     Jbdrawinertia.m draws the inertia 'dumb bells'.

     Jbdrawcircle.m draws circles

   - JBmck.m calculates the linearized equations of motion. It uses the following support module:

     Jbaddb.m adds the mass and inertia of rigid bodies into one equivalent body.

   - Jbvcrit.m calculates the weave and capsize speeds. It uses the following support modules:

     Jbcombint.m combines two interval lists A and B into a new interval list C

Jbpolyint.m makes an interval list A in x where the polynomial p(x) is positive.

- Jbdoubleroots.m calculates double root speed(s), speeds at which a pair of real eigenvalues combine into a complex conjugate pair or visa versa, analytically using MATLAB's Symbolic tool box.
- Jbroots.m calculates double root speed(s) by interpolation from the existing eigenvalues. Because the original algorithm requires MATLAB's Symbolic tool box and can run very slowly, another method was implemented that may not be quite as accurate, but that can run must faster. It takes advantage of the observation that at the speed at which a real eigenvalue pair transforms into a conjugate par of imaginary eigenvalues the shape of the curve on the imaginary plane is that of a parabola and the real part appears linear. First, it iteratively searches for the points on either side of the bifurcation. It then uses MATLAB's polyfit() function to find polynomial coefficients that best fit the parabola and the line. Finally, it returns the value of the forward speed at which these two curves intersect. The results depend on the size of the forward speed step used when calculating eigenvalues and generally match those for the slower algorithm to several digits.
- Jbeig.m calculates the eigenvalues and eigenvectors with MATLAB's eig() function.
- Jbrev.m plots the eigenvalues in appropriate axis on the main Jbike6 window using all the settings and styles specified by the user.
- Jbplot4.m plots the eigenvalues four different ways in a separate window:

1. V vs Re(λ)
2. V vs Im(λ)
3. Re(λ) vs Im(λ)
4. V vs Re(λ) vs Im(λ) in 3D.

This separate window also provides and supports all of MATLAB's plot manipulation controls: pan, zoom, rotate (particularly useful for understanding the 3D plot), etc.

For screenshots, see on-line help appendix on page 43 below.

On-line Help

The on-line help is written in HTML with MS Word and extensively hyperlinked. JBike6 launches it in the default browser for the machine on which it is running when requested by the user.

All support files, mostly digital images, are kept in a subdirectory with the name JBhelp_files.

There is also a separate, much smaller version, extracted from the main file to provide a quick reference just on the parameters.

For complete contents, see appendix on page 28 below.

Web Site

The web site's function is to promote world-wide distribution of JBike6. Toward that end, it has a small footprint to ease access for potential users with slower Internet access or with smaller displays. It describes JBike6 as concisely as possible and provides extensive links for potential users who may be unfamiliar with its terminology.

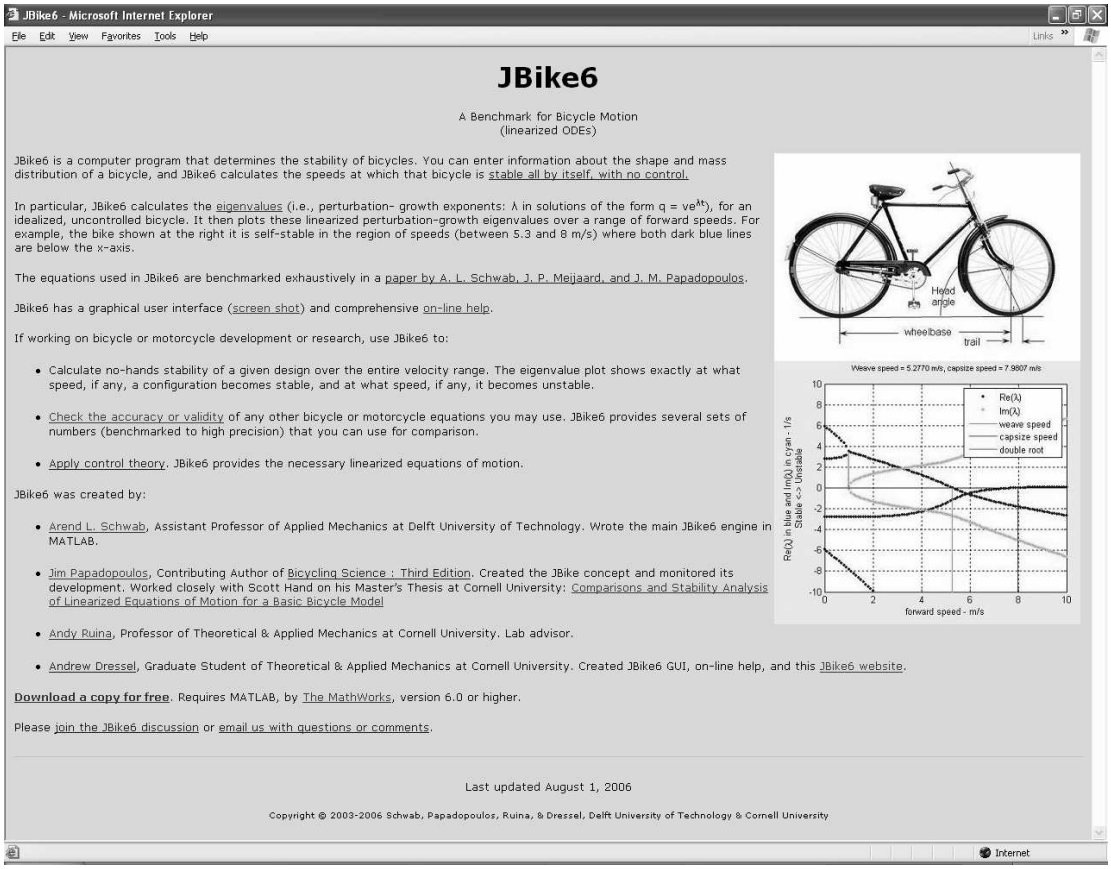
It invites users to download a free copy of JBike6 and prompts for minimal contact information, acceptance of the license agreement, and a brief description of the

intended use. Due to limitations on server functionality, this interactive capability is provided by a third-party web site for a nominal fee. All the captured information is e-mailed to each of the four authors and intended only to help make JBike6 as useful as possible.

The license agreement is designed to prevent abuse of JBike6, protect the intellectual property of the authors, and limit their liability.

The web site also links to an on-line discussion in order to promote communication between geographically disperse JBike6 users. Again, due to server limitations, this interactive functionally is hosted by a third-party web-site.

The main page, as shown below, can be viewed all at one time on a suitably sized monitor.



Additional pages include:

- A description of self-stability with a video demonstration
- A discussion of eigenvalues
- An example of how to benchmark with JBike6
- An example o how to apply control theory to JBike6

Links to external sites include:

- The benchmark paper by paper by A. L. Schwab, J. P. Meijaard, and J. M. Papadopoulos
- Authors individual web pages
- The Mathworks' MATLAB page

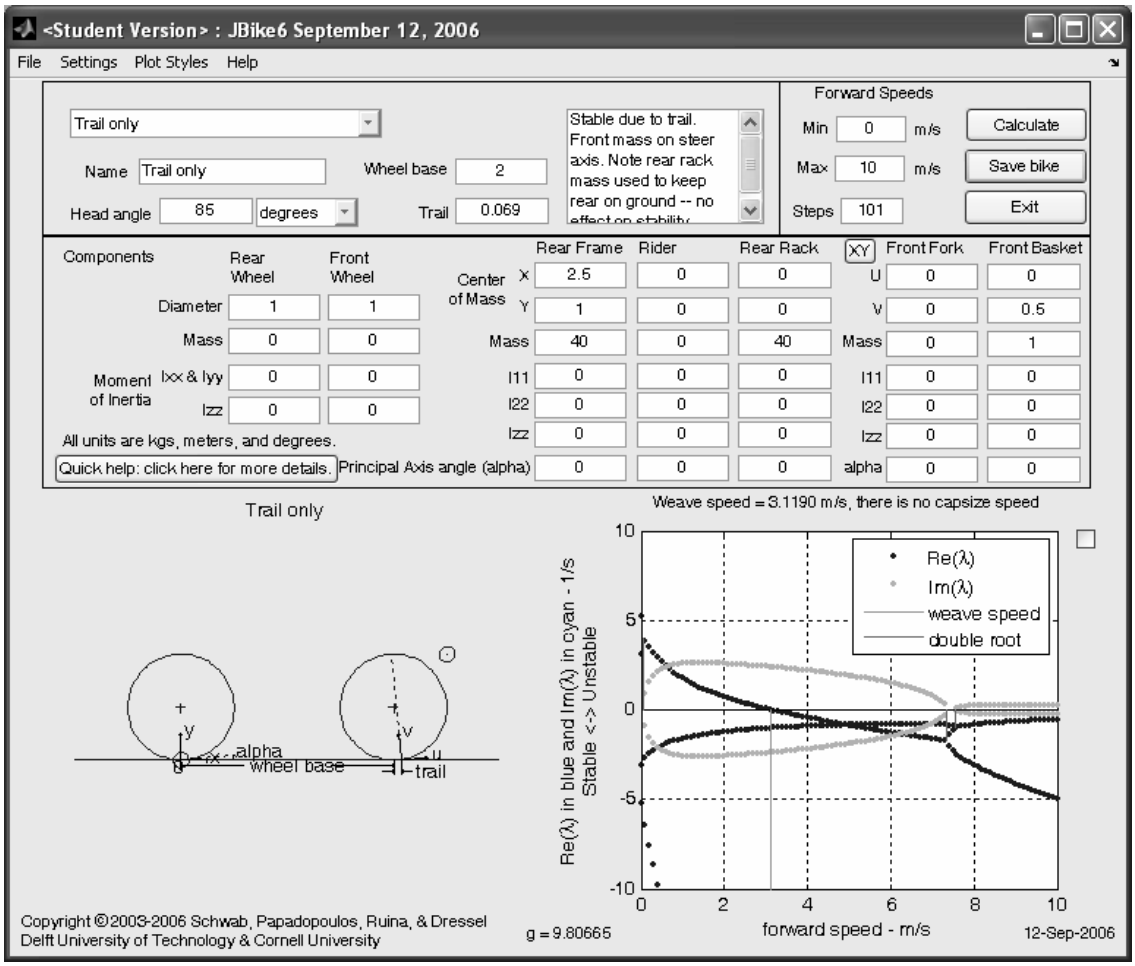
For complete contents, see appendix on page 78 below.

CHAPTER FOUR

CONCLUSIONS

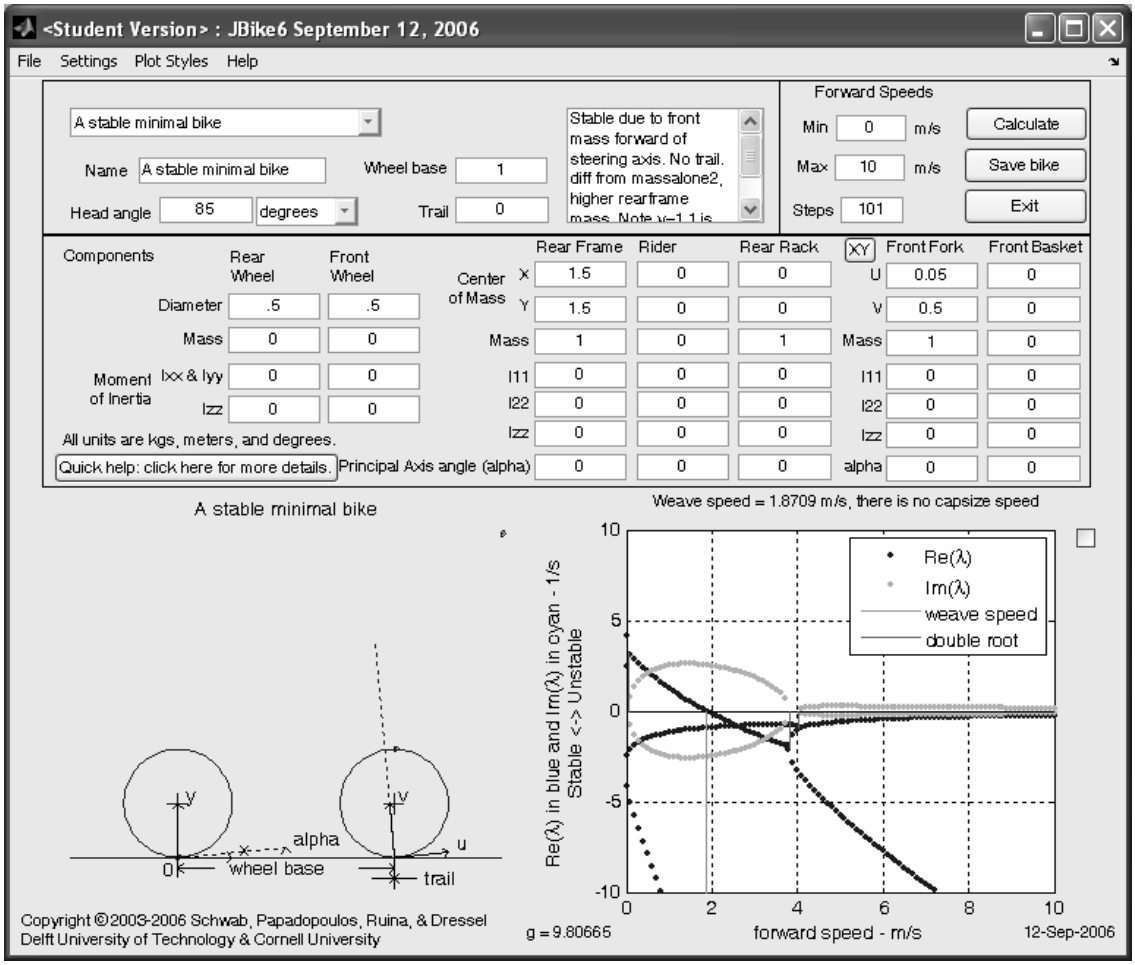
Self-stability without Gyroscopic Effects

Examination of different bike configurations in JBike6 contradicts a conclusion drawn from physical experiment by Jones in his famous 1970 paper. Specifically, although the gyroscopic effects of the wheels are not necessary for riding a bike, they are necessary for self-stability. It is easy to devise a bike with no mass in the wheels that should be self-stable at a range of forward speeds. Below is a screen shot of JBike6 showing one example. Note that the mass and inertia of the wheels are zero.



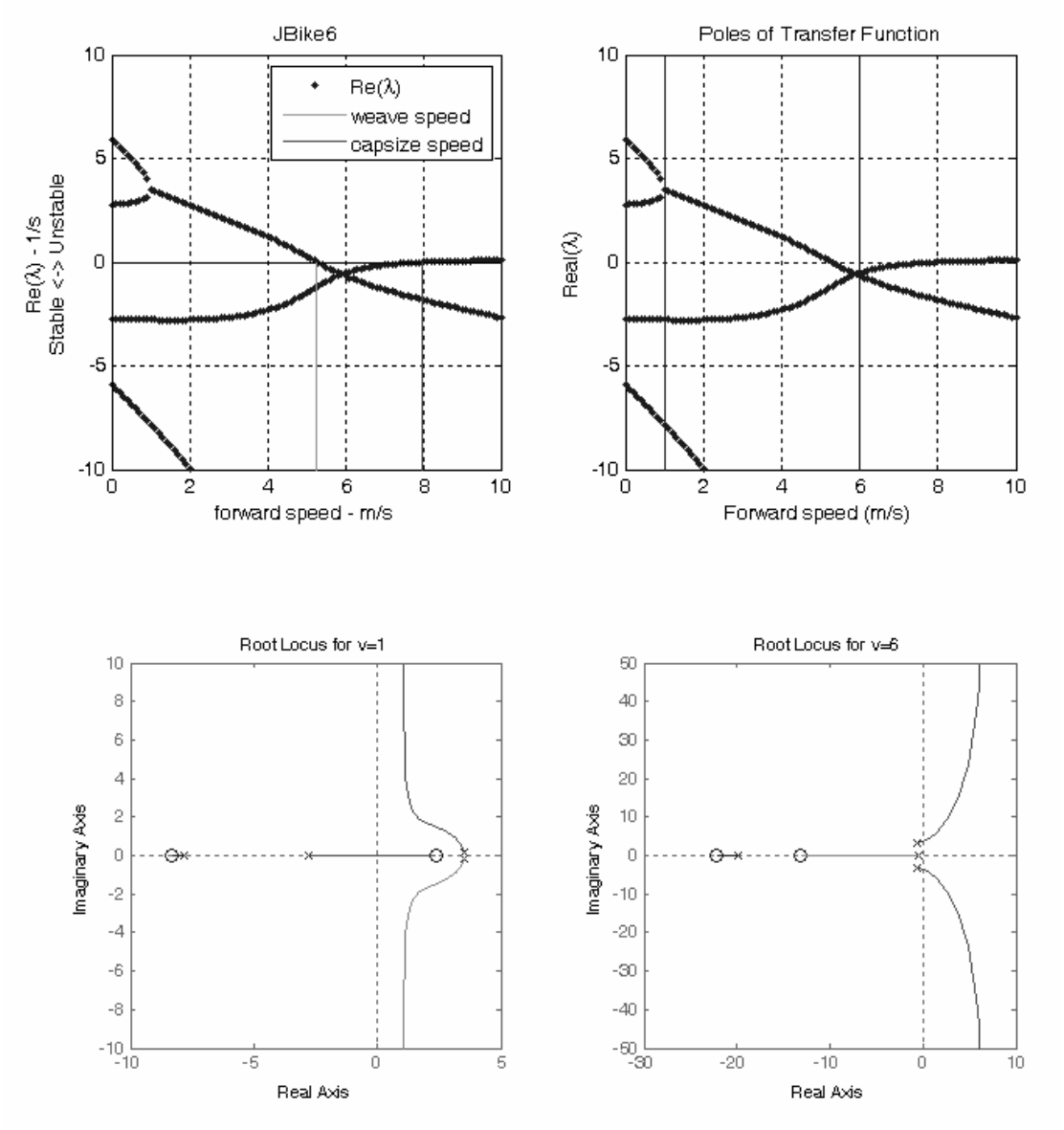
Self-stability without Gyroscopic Effects or Trail

Also contrary to what Jones claims, it is even possible to find a bike with no mechanical trail, a line through the steering axis intersects the ground exactly where the front wheel touches the ground, which also should be self-stable. This is apparently due to the mass of the front assembly forward of the steering axis. The task remains to build such a bike and perform physical experiments. Below is a screen shot of JBike6 showing one example. Note that the mass and inertia of the wheels and the trail are all zero.



Applying Control Theory

JBike6 provides an ideal platform for the application of control theory to bikes. The example on the web site and included below in the appendix shows how to prepare the equations of motion for use with MATLAB's Control System Toolbox. Below is an example image showing the usual JBike6 eigenvalues, poles of the equivalent transfer function for confirmation, and example root locus plots for speeds of 1.0 and 6.0 m/s.



APPENDICES

Details of Derivation and Analysis

Rearrangement of Terms in Hand's Equations

In his formulation, Hand used variables to represent angular momentum due to spinning of the wheels: H_r, H_f, and H_t. This, of course, must incorporate the forward speed of the bike V and thus obscures the symmetry of the expressions for damping and stiffness:

$$C_{\psi\theta} = H_f \cos\lambda + \frac{c_f}{c_w} H_t$$

$$C_{\theta\psi} = -\left(H_f \cos\lambda + \frac{c_f}{c_w} H_t \right) + V\left(T_{yz} \frac{\cos\lambda}{c_w} - \frac{c_f}{c_w} m_t \bar{h}_t \right)$$

$$K_{\theta\psi} = g\upsilon - H_t V \frac{\cos\lambda}{c_w} - V^2 \frac{\cos\lambda}{c_w} m_t \bar{h}_t$$

$$K_{\psi\psi} = -g\nu \sin\lambda + VH_f \sin\lambda \frac{\cos\lambda}{c_w} + V^2 \upsilon \frac{\cos\lambda}{c_w},$$

Instead, I use his definition of forward speed, $V = \dot{Y} = a_r \dot{\phi}_r$, and angular momentum, $H_r = \dot{\phi}_r C_r$, to express angular momentum as a function of forward speed, polar mass moment of inertia, and wheel radius, $H_r = V \frac{C_r}{a_r}$, and substitute it into the expressions above:

$$C_{\psi\theta} = V\left(\frac{C_f}{a_f} \cos\lambda + \frac{c_f}{c_w}\left(\frac{C_f}{a_f} + \frac{C_r}{a_r} \right) \right)$$

$$C_{\theta\psi} = -C_{\psi\theta} + V\left(T_{yz} \frac{\cos\lambda}{c_w} - \frac{c_f}{c_w} m_t \bar{h}_t \right)$$

$$K_{\theta\psi} = g\upsilon - V^2 \left(H_t \frac{\cos\lambda}{c_w}\left(\frac{C_f}{a_f} + \frac{C_r}{a_r} \right) - \frac{\cos\lambda}{c_w} m_t \bar{h} \right)_t$$

$$K_{\psi\psi} = -g\nu \sin\lambda + V^2 \left(\frac{C_f}{a_f} \sin\lambda \frac{\cos\lambda}{c_w} + \upsilon \frac{\cos\lambda}{c_w} \right),$$

Calculation of weave frequency and steer-lean phase lag from eigenvalues and eigenvectors

In the case of complex eigenvalues, $\lambda = \alpha \pm i\omega$, and eigenvectors, $\vec{q}_0 = \begin{bmatrix} \varepsilon_1 + i\beta_1 \\ \varepsilon_2 + i\beta_2 \\ \varepsilon_3 + i\beta_3 \\ \varepsilon_4 + i\beta_4 \end{bmatrix}$, the assumed solution, $\vec{q} = \vec{q}_0 e^{\lambda t}$, takes the form:

$$\vec{q}(t) = C_1 e^{(\alpha+i\omega)t} \begin{bmatrix} \varepsilon_1 + i\beta_1 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} + C_2 e^{(\alpha-i\omega)t} \begin{bmatrix} \varepsilon_1 - i\beta_1 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

$$\vec{q}(t) = e^{\alpha t} \left(C_1 e^{i\omega t} \begin{bmatrix} \varepsilon_1 + i\beta_1 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} + C_2 e^{-i\omega t} \begin{bmatrix} \varepsilon_1 - i\beta_1 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \right)$$

Use Euler's equation: $e^{i\omega} = \cos\omega + i\sin\omega$

$$\vec{q}(t) = e^{\alpha t} \left(C_1 (\cos\omega t + i\sin\omega t) \begin{bmatrix} \varepsilon_1 + i\beta_1 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} + C_2 (\cos\omega t - i\sin\omega t) \begin{bmatrix} \varepsilon_1 - i\beta_1 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \right)$$

$$\vec{q}(t) = e^{\alpha t} \left(C_1 \begin{bmatrix} (\cos\omega t + i\sin\omega t)(\varepsilon_1 + i\beta_1) \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} + C_2 \begin{bmatrix} (\cos\omega t - i\sin\omega t)(\varepsilon_1 - i\beta_1) \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \right)$$

$$\vec{q}(t) = e^{\alpha t}\left(C_1\begin{bmatrix} (\varepsilon_1 + i\beta_1)\cos\omega t + (i\varepsilon_1 - \beta_1)\sin\omega t \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} + C_2\begin{bmatrix} (\varepsilon_1 - i\beta_1)\cos\omega t - (i\varepsilon_1 + \beta_1)\sin\omega t \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \right)$$

$$\vec{q}(t) = e^{\alpha t}\begin{pmatrix} (C_1 + C_2)\varepsilon_1\cos\omega t - (C_1 + C_2)\beta_1\sin\omega t \\ + i(C_1 - C_2)\beta_1\cos\omega t + i(C_1 - C_2)\varepsilon_1\sin\omega t \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}$$

Separate into real and imaginary parts:

$$\vec{q}(t) = (C_1 + C_2)\begin{pmatrix} \varepsilon_1\cos\omega t - \beta_1\sin\omega t \\ \cdot \\ \cdot \\ \cdot \end{pmatrix} + i(C_1 - C_2)e^{\alpha t}\begin{pmatrix} \beta_1\cos\omega t + \varepsilon_1\sin\omega t \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}$$

Introduce phase shift: $a\sin x + b\cos x = \sqrt{a^2 + b^2}\,\sin(x + \phi)$ where $\phi = \tan^{-1}\left(\frac{b}{a}\right)$

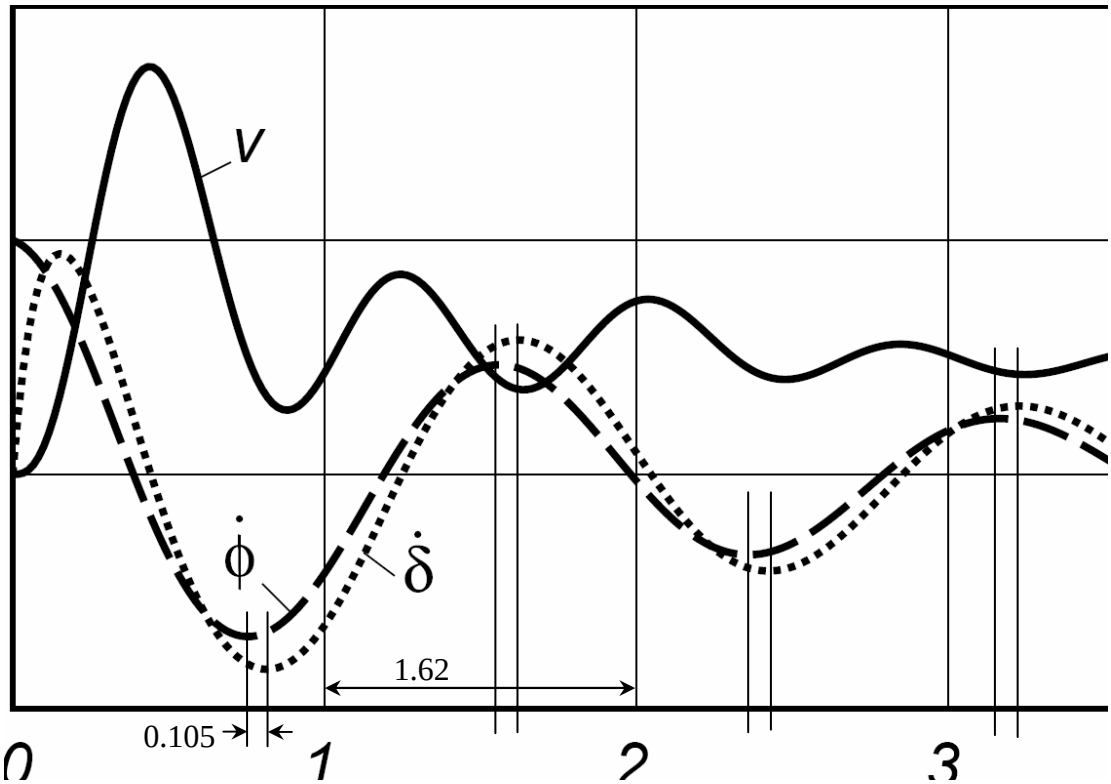
$$\vec{q}(t) = (C_1 + C_2)\begin{pmatrix} \sqrt{\varepsilon_1{}^2 + \beta_1{}^2}\,\sin\left(\omega t + \frac{\varepsilon_1}{-\beta_1}\right) \\ \cdot \\ \cdot \\ \cdot \end{pmatrix} + i(C_1 - C_2)e^{\alpha t}\begin{pmatrix} \sqrt{\varepsilon_1{}^2 + \beta_1{}^2}\,\sin\left(\omega t + \frac{\beta_1}{\varepsilon_1}\right) \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}$$

Finally, recall that $\vec{q}(t) = \begin{bmatrix} \dot{\theta}_r(t) \\ \dot{\psi}(t) \\ \theta_r(t) \\ \psi(t) \end{bmatrix}$, and use just the real part:

$$\begin{bmatrix} \dot{\theta}_r(t) \\ \dot{\psi}(t) \\ \theta_r(t) \\ \psi(t) \end{bmatrix} = (C_1 + C_2) \begin{pmatrix} \sqrt{\varepsilon_1{}^2 + \beta_1{}^2} \sin\left(\omega t + \frac{\varepsilon_1}{-\beta_1}\right) \\ \sqrt{\varepsilon_2{}^2 + \beta_2{}^2} \sin\left(\omega t + \frac{\varepsilon_2}{-\beta_2}\right) \\ \cdot \\ \cdot \end{pmatrix}$$

Confirmation of phase lag calculation:

1. Use plot from Benchmark paper showing lean (roll) rate $\dot{\phi} = \dot{\theta}_r$ and steer rate $\dot{\delta} = \dot{\psi}$ as generated by benchmarked fully non-linear simulation at forward speed of 4.6m/s and initial condition of lean rate of 0.5rad/s.



Benchmark paper reports measured period of 1.60 seconds

Measure phase lag to be ~0.065 seconds (= ~0.105 inches / ~1.62 inches/second).

2. Calculate eigenvalues and eigenvectors with JBike6 for same bike at forward speed v = 4.6 m/s.

```
v =   4.60000000000000

M0 =  80.81722000000001  -2.31941332208709        C1 =                 0 -33.86641391492494
      -2.31941332208709   0.29784188199686                0.85035641456978   1.68540397397560
K0 =  1.0e+002 *
      -7.94119500000000   0.25501260323012        K2 =                 0 -76.59734589573222
       0.25501260323012  -0.07880322817790                             0   2.65431523794604

Vweave = 4.2923825363411057    Vcapsize = 6.0242620153883539

D =
-13.29863951576766    -0.37796622627765 - 3.87284191562967i -0.37796622627765 + 3.87284191562967i -0.62121272697555
V =
  0.00490710380596    -0.66245155444642 - 0.00840242722877i -0.66245155444642 + 0.00840242722877i  0.62121272697555
 -1.00000000000000     0.79840624663021 - 0.20159375336979i  0.79840624663021 + 0.20159375336979i -0.31608462644031
 -0.00036899291842     0.01438690079675 + 0.16964642661838i  0.01438690079675 - 0.16964642661838i -1.00000000000000
  0.07519566184303    -0.07149176328283 - 0.19917796580830i -0.07149176328283 + 0.19917796580830i  0.50881865859250
```

Note entries of M0, C1, K0, and K2 and critical speeds all match benchmark paper to ~12 digits.

Weave frequency:

At v = 4.6, λ_w = 0.37796622627765 - **3.87**284191562967i

Frequency $\omega = \frac{2\pi}{T} = \frac{2*3.14156}{1.622}$ = **3.87**372706977780, which matches frequency predicted by the eigenvalue to three digits.

Phase lag:

First, the value from the plot above: 360°*0.065s/1.6s = 14.6°

Then use $\begin{bmatrix} \dot{\theta}_r(t) \\ \dot{\psi}(t) \\ \theta_r(t) \\ \psi(t) \end{bmatrix} = (C_1 + C_2) \begin{pmatrix} \sqrt{\varepsilon_1^{\,2} + \beta_1^{\,2}} \sin\left(\omega t + \frac{\varepsilon_1}{-\beta_1} \right) \\ \sqrt{\varepsilon_2^{\,2} + \beta_2^{\,2}} \sin\left(\omega t + \frac{\varepsilon_2}{-\beta_2} \right) \\ . \\ . \end{pmatrix}$, derived above.

REMEMBER! In JBike6, the source of these numbers, lean and steer have opposite sense/orientation, so need to negate values corresponding to one or other and then subtract the phase lag of $\dot{\theta}_r$ from $\dot{\psi}$

```
(atan2(0.66245155444642, -0.00840242722877) –
 atan2(0.79840624663021, 0.20159375336979))*180/pi = 14.9°
```

JBike6 Help
(September 12, 2006)

Installing and Running JBike6

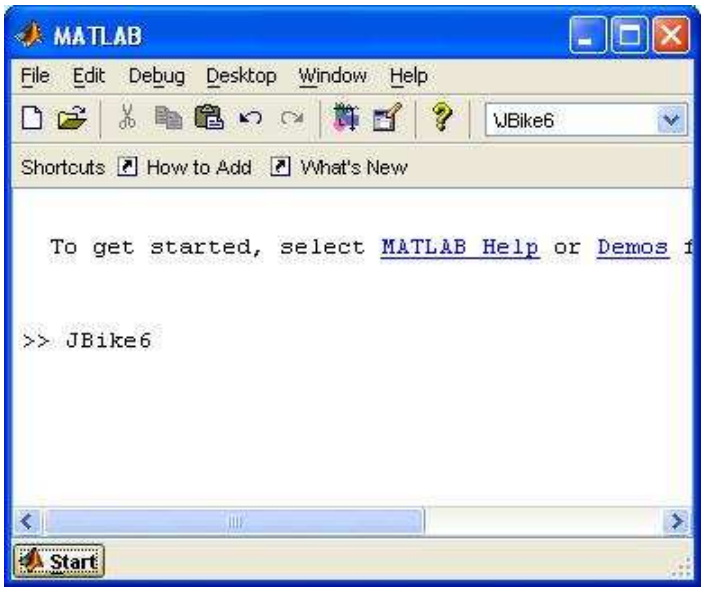
Installing

Download the distribution file onto your computer.

Unzip the distribution file into a subdirectory on your computer.

Running

Start MATLAB from The Mathworks. The JBike6 GUI (Graphical User Interface) requires MATLAB version 6.0 or higher. To run without the GUI see the section below on Running JBike6 without the GUI.



Tell MATLAB where to find JBike6 by either:

1. In MATLAB, make the current directory the directory into which you unzipped JBike6. You can do this through the MATLAB desktop or with the cd command in the MATLAB command window.

2. Alternatively, if you have MATLAB version 7.0 or higher, you may add the JBike6 directory to the MATLAB path. You can do this through the 'Set Path…' option of the 'File' menu or with the path command in the MATLAB command window.

Run the file JBike6.m by either:

1. typing ‘JBike6’ on the MATLAB command line and pressing the ‘Enter’ key on your keyboard, or

2. opening the JBike6.m file in MATLAB’s Editor and selecting ‘Run’ from the ‘Debug’ menu or pressing the ‘F5’ key.

In fact, if you run JBike6 from MATLAB’s Editor, you may skip telling MATLAB where to find JBike6, and it will prompt you to let it automatically either change the current directory or add to the path.

See the MATLAB help for more details.

Running without the GUI

If there is some functionality you need that the GUI (Graphical User Interface) does not offer, or if the GUI is getting in your way, you can run JBike6 as a command-line application. The first thing you need to do to make this possible is edit the JBike6.m file and change the line that reads:

```
use_GUI = 1
```

to

```
use_GUI = 0
```

Then simply run JBike6 as before, and now it will not launching the GUI. JBike6 reads from the JBini.m file all the parameters that you would specify in the GUI.

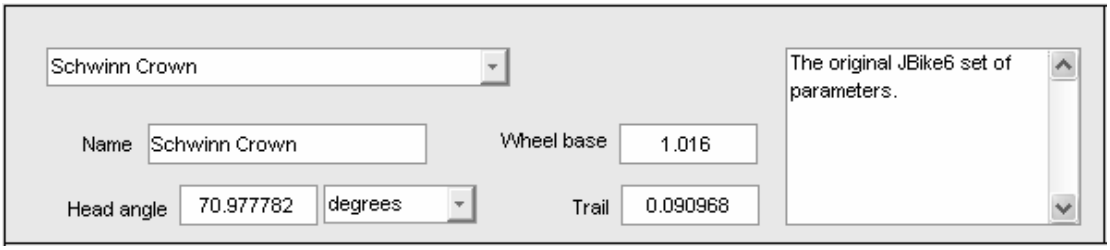
The JBini.m file contains comments to explain the variables, their coordinate systems, and their units. When you run JBike6 without the GUI, it draws the same bike diagram and eigenvalue plot as with the GUI, but in separate windows.

For more details about running JBike6 without the GUI, see the Appendix below.

Using JBike6

You can specify the mass, location, orientation, and inertia for all the bodies that make up a bike, or you can load a previously saved parameter set from the pull-down menu in the upper left of the window.

You can save all the parameters that describe a particular bike configuration and retrieve them for later use. You may also simply discard them if they do not prove interesting or useful.

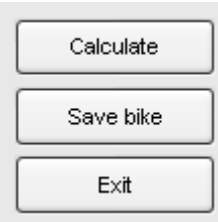


Selecting an existing bike design

To select an existing bike design, provided with JBike6 or previously saved by you, simply select one from the pull-down list in the upper-left corner of the main window.

JBike6 will display all of its parameters, draw a stylized diagram of it, and calculate its eigenvalues and plot them.

Calculating and plotting eigenvalues



To calculate and plot the eigenvalues for a bike, click on the '**Calculate**' button in the upper right of the window. Jbike6 reads all the current bike parameters you have specified, checks them for validity, draws the bike, calculates the eigenvalues, and plots them.

There is an option in the 'Settings' menu to turn on and off the display of status messages in MATLAB's command window while calculating eigenvalues. Turn this on to see what JBike6 is doing while you wait. Turn this off to save a little time during each calculation.

Specifying a new bike design

There are two ways that you can specify a new bike design:

1. Start from scratch: select 'Blank' from the pull-down list of bikes. All the parameter values are zero. Type a new name and all the parameter values that you want. Then save the new design. JBike6 cannot work with a design you enter from scratch until you save it.

2. Start with an existing design: select any pre-existing design from the pull-down list of bikes. Type a new name and change all the parameter values that you want.

Each number may contain any of the digits 0 through 9, a decimal point, a leading + or - sign, and an 'e' or 'd' preceding a power of 10 scale factor: for example the ASCII text string ‘-1.23e2’ represents the value -123. Read more about precision below.

Until you press the 'calculate' button, all the text-entry controls, such as ‘trail’ and ‘mass’ contain whatever and exactly what you type in there. As soon as you press the 'calculate' button, or otherwise start JBike6 on the task of calculating and plotting the eigenvalues, it reads the contents of each control and uses MATLAB’s internal ‘str2num’ function to convert it to a an IEEE 64 bit floating point number. That function returns a number if it can, otherwise it returns nothing: the ‘length’ of the variable contents is actually zero.

The calculations required to generate eigenvalues cannot work with variables that contain nothing, and they would report an error if they did, so JBike6 checks for this condition, and when it occurs, sets the value to the 64 bit representation of zero. Then, for completeness, it puts back into that control the ASCII text string representation of that number. So, any field that did not have a valid number, including fields that were blank, get set to '0'.

Saving bike parameters

You can save all the parameters of a particular bike design either with the ‘Save bike’ button, the ‘Save bike parameters’ option of the ‘File’ menu, or by pressing CTRL-S (the ‘Control’ key and the ‘s’ key simultaneously).

When you select ‘Save bike‘, JBike6 writes an ASCII text configuration file that contains all the parameters necessary to describe the bike. The file name always begins with ‘BJini’, contains the bike name that you specify, and ends with an ‘.m’ extension. In order to make the file name usable by JBike6 the next time it starts, it removes periods and parenthesis and replaces blanks with underscore characters. Also to be a valid file name in the MS Windows file system it cannot contain any of the following characters: \ / : * ? ” < > |, so JBike6 removes them as well.

JBike6 also saves the current gravity constant g and calculated critical speeds in this file, if they exists, but only as comments and will not read them back in. Each time you select a different bike or select ‘Calculate’, JBike6 uses the current value of g, which you can specify, to calculate eigenvalues and the weave and capsize speeds.

When JBike6 starts, it reads all the configuration files it can find and sorts them alphabetically to build the list of available bikes for the pull-down menu. The directory from which JBike6 reads and to which it writes bike configuration files defaults to the JBini_files subdirectory of the program directory: where the JBike6 program files are located. You may change this directory by selecting '**Set 'save bike' directory…**' from the 'Settings' menu.

Discarding bike parameters

You may discard changes you make to bike parameters if they do not prove interesting or useful. To discard them, simply select a bike (the same one or a different one) from the pull-down menu or exit JBike6.

Saving settings and styles

There is an option in the 'Settings' menu to save all the other settings, such as the forward speeds, the gravitational constant, etc., and styles when JBike6 exits. If you have this setting checked, JBike6 writes all your settings to a configuration file, JBconfig.m. The next time you start JBike6, it will read this file and restore all your settings and styles. Note that JBike6 can only save your settings and styles if you exit by either the 'Exit' option in the 'File' menu or the 'Exit' button. If you exit with the 'close window' button at the upper right of the window, MATLAB does not give JBike6 an opportunity to save your settings and styles.

Saving calculated eigenvalues

There is an option in the 'File' menu to save the calculated eigenvalues along with their corresponding speeds to a file. When you specify the file name, you can specify either a MATLAB *.mat file or an ASCII text file. After you specify the file name and type, JBike6 writes the eigenvalues and their corresponding speeds to that file in the appropriate format.

You can open a MATLAB *.mat file with the 'load' command at the MATLAB command prompt. See the MATLAB help for more details. You can open an ASCII text file in Notepad, Microsoft Excel, or similar application.

Exiting JBike6

There are several ways to exit JBike6:

1. Select '**Exit'** from the 'File' menu.

2. Press **CTRL-X** (the 'Control' key and the 'x' key simultaneously).

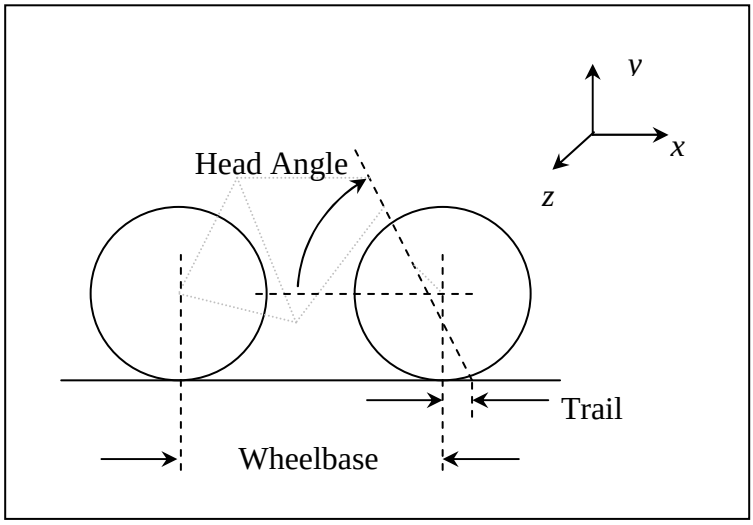
3. Click on the **'Exit' button** in the upper right-hand corner of the main window.

4. Click on the 'Close Window' button n the extreme upper right-hand corner of the main window.

5. Exit MATLAB, by any of its various methods. See the MATLAB help for more details.

Caution: if you have checked the 'Save settings and styles when exiting' option in the 'Settings' menu, and you exit JBike6 by any method other than 1, 2, or 3, above, then MATLAB does not give JBike6 an opportunity to save styles and settings. Any changes you have made to settings or styles will be lost.

The Parameters



You specify a particular bike configuration by entering certain parameters. For the idealized model of a bike that JBike6 uses, the parameters that you can enter on the main window provide all the information that JBike6 needs to calculate the eigenvalues.

All mass is in kilograms, distance in meters, and speed in meters per second.

You may toggle between angles in degrees or radians by selecting either one from the pull-down menu next to the head angle. JBike6 automatically recalculates the values of all the angle parameters to match the units you choose.

JBike6 treats all parameters as decimal expansions to 16 significant digits. In other words, 72 degrees really means 72.00000000000000 degrees, even though trailing zeros are not shown. Read more about precision below.

The parameters you enter can not be arbitrary. As described below, certain ones have to be nonzero or positive. In order for the model to make physical sense, you have to limit certain ones: keep the mass above the ground, keep wheels from bumping each other, etc.

When JBike6 displays values in its own window, it may truncate some, such as the critical speeds displayed above the eigenvalue plot, in order to fit in the available space, but it displays values in the MATLAB command window with all 16 available digits. Read more about precision below.

JBike6 displays ‘tooltips’ when you hover the cursor over any of the parameter input controls.

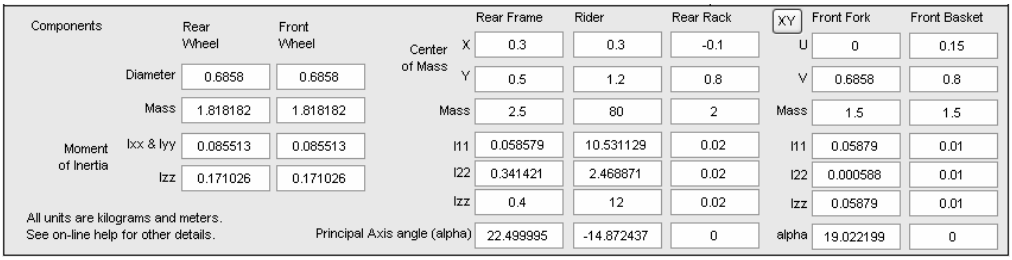
Bike Parameters

The **wheelbase** is the horizontal distance between the hub (or ground contact point) of the front wheel and the rear wheel. The wheelbase can be positive or negative, but not zero, and the magnitude should be greater than sum of wheel radii in order for bike to be physically possible.

The **head angle** is the angle the head tube (steering axis) makes with the horizontal. A 90° head angle would be vertical. By convention, positive angles are clockwise from the horizontal. Common head angles for road bikes range from 70° to 75°. JBike6 accepts any angle, but angles close to 0° are not practical for steering.

The **trail** is the horizontal distance from the front hub (or ground contact point) to the point where the steering axis intersects the ground. By convention, positive trail is when the front wheel contact point is behind, toward the rear wheel, the steering axis intersect point.

Component Parameters



Rear: The **center of mass** of the rider, rear frame, and rear rack are all measured in the **xy-coordinate system**. The **origin** is located at the rear wheel contact point, the **x-axis** is horizontal forward, the **y-axis** is

vertical up, and the **z-axis** is out of the plane following the right-hand rule.

The wheelbase and wheel diameters completely specify the locations of the wheel centers of mass.

Front: You can specify the center of mass of the front fork and the front basket in either the **uv-coordinate system** or the same xy-coordinate system as the rear of the bike. You can toggle between these two coordinate systems by clicking on the XY or UV button at the upper left of the front fork parameters. The labels below the button indicate which coordinate system JBike6 is currently using. The origin of the uv-coordinate system is located at the intersection of the steer axis with the ground and the coordinate system is rotated so that the **v-axis** is along the steering axis: rotated from the y-axis counter clock-wise by $\pi/2$ radians (90°) minus the head angle. (See the bike diagram below.) JBike6 automatically recalculates the values of front fork and basket coordinates to match the coordinate system you choose. JBike6 uses the uv-coordinate system internally, and saves (and reads) bike configurations with the uv-coordinate system.

For the **inertia tensor** of each mass, the **1-axis** is rotated counter-clock-wise from the horizontal by the **principal axis angle** (alpha) for that mass, and the **2-axis** is rotated in the same direction from the vertical by the same amount. (See the bike diagram below.) The principal axis angle for the front fork and the front basket is always measured counter clock-wise from the axes of the xy-coordinate system, even if their center of mass is specified in the uv coordinate system.

I_{11} represents the inertia about the 1-axis, I_{22} the inertia about the 2-axis, and I_{zz}, about the z-axis.

Note that the wheels are treated as symmetrical about the z-axis, and so $I_{xx} = I_{yy}$.

Before JBIke6 begins calculating eigenvalues, it combines the rear frame, rider, and rear rack into one single rigid body, and it combines the front fork and front basket into a second single rigid body. This is necessary because of the simplified bike model used to derive the equations of motion that JBike6 uses.

For further discussion see the More on Parameters section in the appendix below.

Forward Speeds

The actual forward speed of the bike is a crucial component of the eigenvalue calculations. In order to see how eigenvalues change with speed, you can enter a range of speeds to use. You can specify a minimum and maximum speed, and the number of individual speeds (steps) to use in between.

Gravity

You can change the gravitational acceleration constant (small 'g') used in the calculations by selecting the 'Set gravity constant…' option of the 'Settings' menu. For your convenience, the currently set value of g is displayed at the bottom of the window. You may enter any positive value for gravity that you want. JBike6 will calculate different weave and capsize speeds for different values of g.

It can be shown that vastly different values (for example, g on the moon) drastically change the zero-speed eigenvalues; but at high speed only the capsize eigenvalue (value: nearly zero) is noticeably affected. In other words, high speed bike dynamics are largely independent of gravity.

Read more about Gravity in the Appendix below.

Precision

In general, you can treat all inputs as having 16 significant digits. It does not matter whether you enter 3, 3.0, or 3.00000000000000. JBike6 stores all of them the same way internally.

A change near the 7^{th} digit of the trail parameter produces a change near the 7^{th} digit of the calculated weave speed, a change near the 8^{th} digit produces a change near the 8^{th} digit, and the 9^{th} to the 9^{th}, etc.

Finally, the comparison used to verify the results from JBike6 is able to match 14 digits of the values that JBike6 produces with those from other methods.

Read more about Precision in the Appendix on page 56 below.

Parameter Limitations

Wheelbase: can be positive or negative, but not zero, and magnitude should be greater than sum of wheel radii in order for bike to be constructable.

Head angle: any angle, but angles close to 0° are not practical f or steering. JBike6 produces identical results for values ±360, when using the uv-coordinate system, and ±180 when using the xy-coordinate system.

Trail: any value†

Wheel diameters: must be positive.*

Mass: must be greater than or equal to zero. At least one mass in the front end (fork, basket, or front wheel) must be greater than zero. At least one mass in entire bike must be off the ground (mass greater than zero and y or v coordinate greater than zero). When the ratio between masses approaches 1e20, the determinant of the mass matrix becomes zero, and JBike6 is unable to continue.

Inertia: each value must be greater than or equal to zero and $I_{11} + I_{22} \geq I_{zz}$, $I_{zz} + I_{11} \geq I_{22}$, and $I_{22} + I_{zz} \geq I_{11}$.

Principal Axis Angles: any value.†

Speeds: any values, including negative, so long as min and max are not the same.

Speed steps: at least 2, or 3 in order to find double root speeds via interpolation method.

Gravity: must be greater than zero.*

See Appendix for additional guidelines.

* greater than zero actually means the smallest positive value, 4.9407e-324, or greater.

† You may use scientific notation, with the letter e to specify a power-of-ten scale factor, and a maximum of 16 digits of precision, to enter values: 1e1 = 10. The smallest magnitude is 4.9407e-324, and the largest is 1.7976e+308

Errors

JBike6 does check all the parameters

you enter to make sure that all the requirements listed above are met. Never-the-less, there may be unforeseen parameter combinations for which JBike6 or the underlying MATLAB are unable to calculate the eigenvalues. For example, when the determinant of the mass matrix is zero. In that case JBike6 will report an error, as seen above. You may simply change the parameters and try to calculate the eigenvalues again.

The Bike Diagram

JBike6 draws a representation of the bike you specify in the lower left half of the window. It draws the ground plane, the steering axis as a dotted line, and stylized representations of the wheels, frame, fork, rider, rear rack, and front basket.

It uses stylized dumbbells to illustrate the inertia parameters of each body. You can think any mass you specify as being divided into 6 equal-mass disks (represented by circles, only 5 of which are visible in 2D) located on the ends of three mutually-orthogonal dumbbell arms, oriented along the principal inertia axes. The size of the circles represents the mass with a radius of

$$\sqrt{\frac{mass/6}{400*\pi}}.$$

The length of the arms represents how far the mass is distributed from the center of gravity. Each body has its own Principal Axis angle (alpha) to specify how the I_{11} axis is rotated (counter clock-wise) from the always-horizontal x axis.

The farther away the disks are from the center along the 1-axis and the 2-axis, the more inertia the object has about the z-axis. The larger the value of any one of the three, up to the limits described above, the farther away the disks will be along the other two axes.

The depth of the wheel rims represents the distribution of the wheel mass away from the center of gravity. The deeper the rim, the more inertia the wheel has about the z-axis: its hub. The outer circle is at the actual rim diameter that you specify. The inner circle is at a radius of

$$\sqrt{radius^2 - \frac{mass}{400*\pi}}.$$

Note, it is OK to have a body with large inertia and small mass – even though the dumbbell is drawn as penetrating the ground. It could, in fact, be due to a very tall, lightweight pole extending upward, plus a large mass extending only slightly downward.

See appendix on page 57 for further explanation.

In the ‘Settings’ menu, you can toggle on or off the display of the coordinate axes and labels in the bike diagram.

The Eigenvalue Plot

After calculating the eigenvalues for each forward speed, JBike6 plots the real component of them against the forward speed in the lower right half of the window with dots.

Critical Speeds

In addition simply calculating and plotting the eigenvalues for a range of forward speeds, JBike6 calculates some critical speeds at which the bike behavior undergoes a qualitative change.

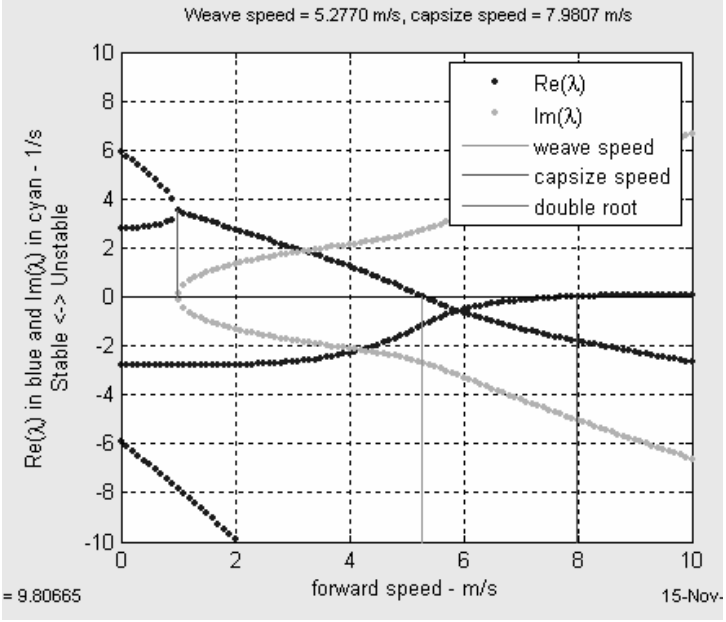
JBike6 also displays and labels these values in the MATLAB command window.

Weave Speed

The forward speed at which oscillatory motion neither grows nor decays. This is represented on the plot by a vertical line. Oscillatory motion is that motion in which the bike leans and steers back and forth.

At speeds below this weave speed, the oscillations will grow until the bike falls over. There may not be a weave speed for a particular bike configuration.

Weave is a sinuous motion of the bike, steering and leaning left and right. When it leans left it also steers left, more than enough to simply return to the upright. The oscillation magnitude is steady at the weave speed, and grows below or shrinks above.



A conventional bike has a single nonzero speed interval of stability, and the weave speed is the lower boundary of this interval. Thus an uncontrolled bike that slows just below the weave speed becomes unstable by weaving. But, if the weave speed is above the capsize speed, there probably is no stable interval.

Capsize Speed

The forward speed at which capsize motion neither grows nor decays. This is represented on the plot by a vertical line. Capsize motion is that motion in which the bike slowly leans and steers, without oscillation, farther and farther until it falls over.

At speeds above this capsize speed, the capsize motion will progress until the bike falls over. There may not be a capsize speed for a particular bike configuration. Odd designs have been found which have no finite capsize speed, and are stable at all speeds above the weave speed.

In capsize, the bike leans and steers to the same side, but the steer is never sufficient to balance or correct the lean. At the capsize speed it just travels in a circle without uprighting. Above the capsize speed, it spirals tighter and tighter (leaning and steering more and more.) Capsize generally takes many seconds until full collapse.

For more information on eigenvalues, stability, and how to interpret this plot see the sections of eigenvalues and interpretation of results below.

Double Root Speeds

The speeds at which **double roots** occur, where two real roots start forming a complex conjugate pair of roots or vice versa, are marked by vertical lines. This is where oscillatory behavior begins, at a frequency that grows rapidly as the speed is surpassed.

JBike6 offers two methods for finding these roots:

1. analytical: using analytical manipulations supplemented by polynomial root finding. This can take several minutes.

2. interpolation: by interpolating from the calculated eigenvalues. This can take several seconds.

There is an option for you to select which method to use and an option to toggle this functionality on and off .in the ‘Settings’ menu.

There is an option in the ‘Settings’ menu to toggle on and off the display of the imaginary components of the eigenvalues. This can be useful in understanding the frequency of the oscillatory motion.

Note that changes you make in the ‘Settings’ menu do not take effect until you calculate and draw the eigenvalues again.

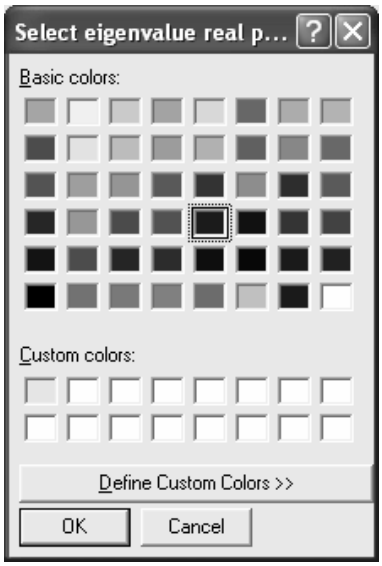
Plot Styles

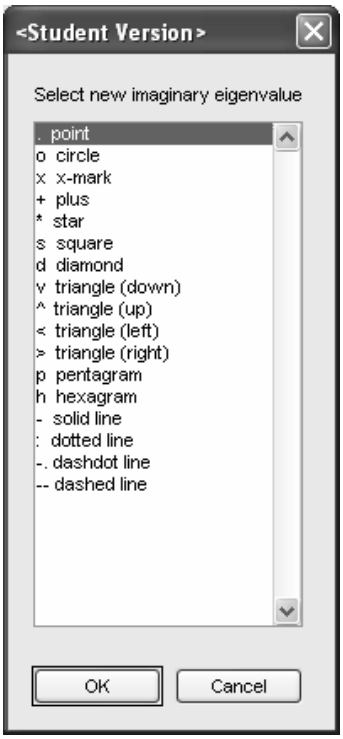
You can change the colors, sizes, symbol types, and line types that JBike6 uses to plot the eigenvalues and critical speeds. Select the appropriate option from the ‘Plot Styles’ menu and then pick the color, enter the size, or pick the symbol or line type you want. You can also set the plot background color.

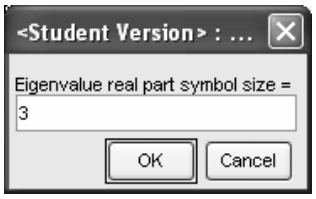
You can also select ‘Black and white styles’ to use a predefined set of black and white colors and appropriately differentiated set of symbol types and line types suitable for black and white output.

You can quickly increase or decrease the sizes of all the plot styles with the ‘Bigger styles’ and ‘Smaller styles’ menu options respectively.

Finally, you can select ‘Reset default colors’ to return the plot windows to the original colors, sizes, symbol types and line types.







There is an option in the ‘Settings’ menu to toggle on and off the display of a legend. You may click on and move the legend from its default position if necessary. There is also a context menu with style options that you can access by right-clicking on the legend. See the MATLAB help for further information about manipulating a legend.

Note that changes in the ‘Plot Styles’ menu do not take effect until you calculate and draw the eigenvalues again.

Additional eigenvalue plots

There is also an option in the ‘Settings’ menu to toggle on and off the display of an additional window with four different plots:

1. V vs Re(λ)
2. V vs Im(λ)
3. Re(λ) vs Im(λ)
4. V vs Re(λ) vs Im(λ) in 3D.

You can use the MATLAB magnify, pan, and rotate tools on these plots along with the rest of MATLAB’s plot manipulation tools. See the MATLAB help for more details.

In the 3D plot, the imaginary eigenvalue components (representing oscillation frequency) are plotted more conventionally: perpendicular to the real eigenvalue components.

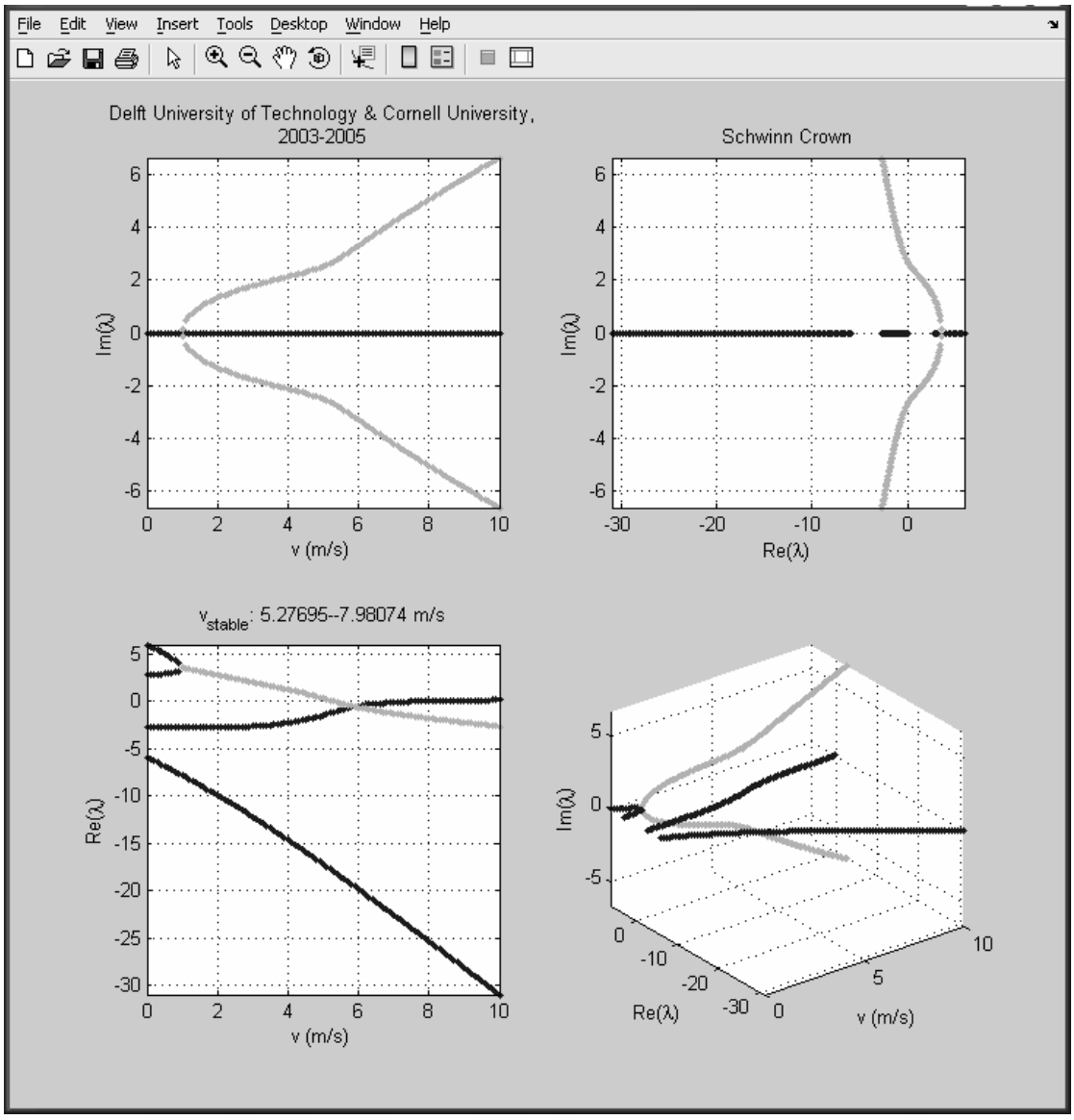
There is an option in the ‘Settings’ menu to use the styles from the single plot in the main window in these 4 plots additional plots. Select the ‘Use Styles menu styles in additional plots’ to have JBike6 draw the real and imaginary parts of the eigenvalues with the styles you have specified in the ‘Plot Styles’ menu.

JBike6 applies colors to the real and imaginary parts slightly differently than in the single eigenvalue plot in the main window.

- In the single eigenvalue plot in the main window, JBike6 plots the real components of all the eigenvalues in one color, and, if you have the ‘Draw imaginary parts of eigenvalues’ option checked in the ‘Settings’ menu, the imaginary components in a different color on the same axis.

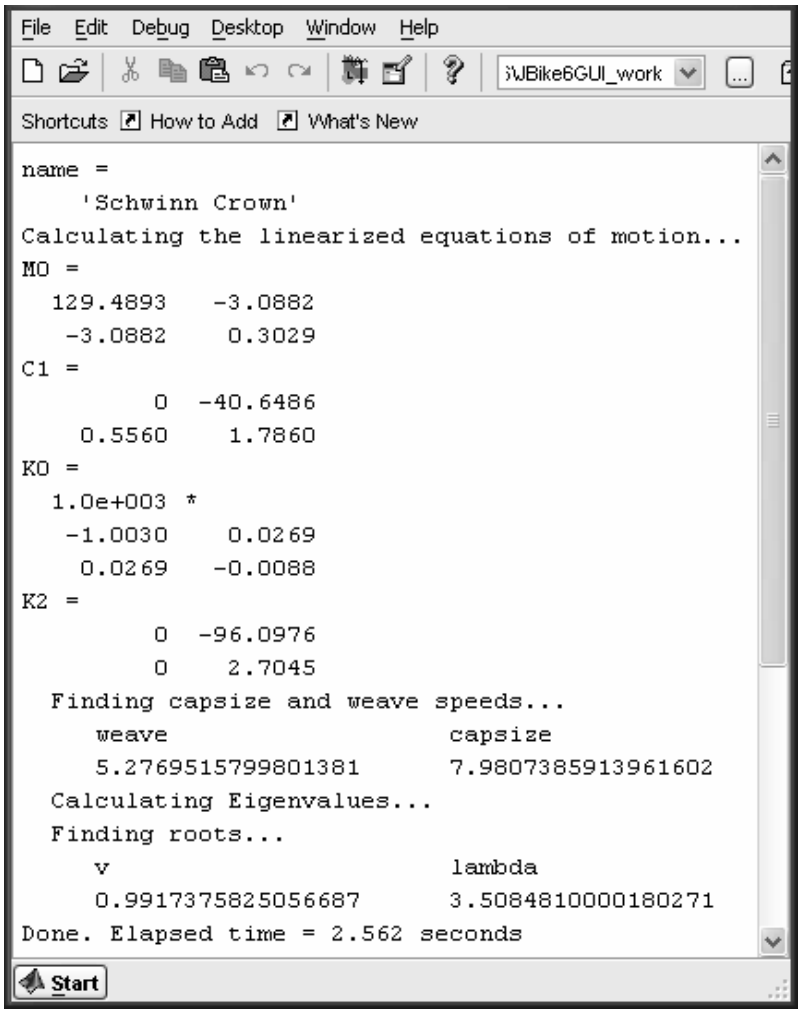
- In the additional, four-plot window, JBike6 always plots real and imaginary components on different axis. It also, if you have the ‘Use Styles menu styles in additional plots’ option checked in the ‘Settings’ menu, uses the ‘real’ color only when an eigenvalue does not have an imaginary component, and it plots all complex eigenvalues with the ‘imaginary’ color.

See screen shot below.


File Edit View Insert Tools Desktop Window Help
Delft University of Technology & Cornell University, 2003-2005
Schwinn Crown
Im(λ)
v (m/s)
Re(λ)
v_stable: 5.27695--7.98074 m/s

MATLAB Command Window Values

As JBike6 calculates certain values, weave speed and capsize speed for example, it can also display these, to all possible 16 digits, in the MATLAB command window.



This can slightly slow the entire operation, and so there is an option in the 'Settings' menu to turn this display off and on.

The full list of values displayed includes:

The bike name

The mass matrix: M0

The damping matrix: C1

The stiffness matrix, in two pieces that are added together before calculating eigenvalues: K0 and K2

The weave and capsize speeds

The eigenvalues and speeds at which double roots appear

The elapsed time

You can plug the values from the mass, damping, and stiffness matrices into the equations of motion given below to perform further analysis such as integrating them forward in time to evaluate lean and heading due to steering input.

Background and Theory

The model

JBike6 uses an idealized rigid-body dynamics model of the bike, defined as four interconnected rigid bodies interacting with a horizontal ground plane:

1. A rear wheel with left-right mass symmetry and polar mass symmetry, rolling on a knife edge, and revolutely connected to the rear frame.

2. A rear-frame, rider, rear rack, all in one rigid assembly, with left-right mass symmetry, revolutely connected to the front fork. (Although JBike6 uses only the entire assembly's mass and inertia, you may separately define a frame, rider, and rack in case you have data only for the separate sub-components.)

3. A front-fork, handlebar, and front basket, all in one rigid assembly, with left-right mass symmetry, revolutely connected to the front wheel.

4. A front wheel with left-right mass symmetry and polar mass symmetry, rolling on a knife edge.

In the complete bike assembly, when held upright and steered straight, all components share the same symmetry midplane. The wheel axles are normal to that plane and the steer axis is in that plane.

All the revolute joints are frictionless, and the rolling occurs without dissipation or slippage.

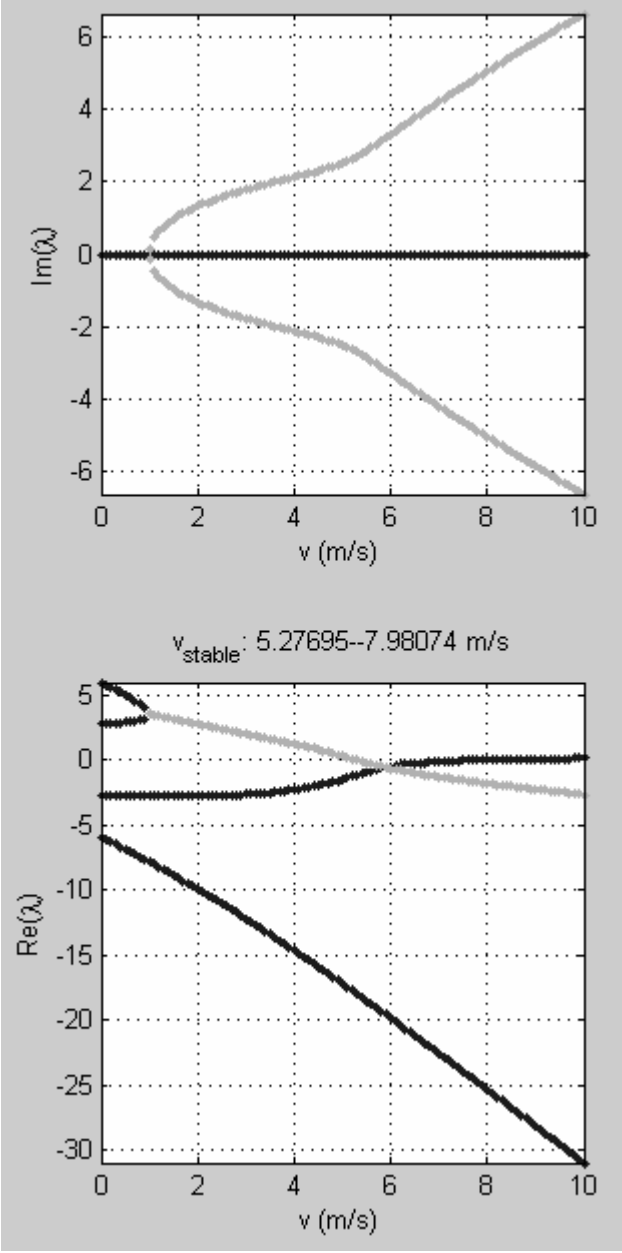
There is no propulsive or braking torque, and no slope or air resistance.

JBike6 uses a linearized analysis for small deviations from the upright straight condition. You should expect perfect accuracy only when steer and lean angles are infinitesimal, however you may expect that many results and trends [easy to find because the governing equations are linear] also hold quite well for realistically large angles. Determining the

inaccuracy resulting from realistic angles is the province of nonlinear analysis.

Note that in a linearized analysis near the upright straight condition, the bike's full geometry and mass distribution is not needed – for example, there is no 'pitch' motion of the rear frame assembly, so that component of inertia is irrelevant. Also, apart from establishing the rotation rate of the wheels, wheel diameter is unimportant – one could as easily envision 'infinitesimal' wheels placed at the bike's two contact points. JBike6 takes full data as input so that they can be unambiguously available for nonlinear analysis.

The two degrees of freedom are q=[lean angle, steer angle], with lean to the right (clock-wise when viewed from the rear) and steer to the left (counter-clock-wise when viewed from above) as positive when driving forward.



Eigenvalues

Definition

Eigenvalues are defined, as the scalar quantities λ such that for matrix [A], if there is a non-zero vector $\vec{x}$ for which this equation $[A]\vec{x} = \lambda\vec{x}$ is true, then $\vec{x}$ is an eigenvector of [A] and λ is its corresponding eigenvalue. Another way to think of them is that if the matrix [A] is a transformation, then the eigenvectors of [A] are those vectors whose directions are not changed by that transformation, and the

corresponding eigenvalues are how much the magnitudes or lengths of those eigenvectors are changed.

Read more about Eigenvalues in the Appendix below.

Interpretation

To see where eigenvalues come into play in the bike system, represent it (see linearized equations of motion below) in state-space form as $\dot{\vec{x}} = [\mathbf{A}]\vec{x}$ (where $\vec{x}$ is a vector that represents all the coordinates and their speeds necessary to describe the system, $\dot{\vec{x}}$ is a vector that represents their time derivatives: speeds and accelerations, and $[\mathbf{A}]$ is a matrix that incorporates all the necessary parameters), and assume a solution to the system of differential equations to have the form $\vec{x} = e^{[\mathbf{A}]t}$ where λ are the eigenvalues of the matrix $[\mathbf{A}]$, *t* is time, and *e* is the base of the natural logarithm**.**

Exponential growth or decay

Then, for any positive values of *λ*, $e^{[\mathbf{A}]t} > 1$, the solution $\vec{x}$ will grow. For negative values of *λ*, $e^{[\mathbf{A}]t} < 1$, the solution $\vec{x}$ will shrink. A growing solution moves away from the initial state and so is **unstable**, and a shrinking solution returns to the initial state and so is **stable**.

Thus the eigenvalues, *λ*, of the matrix [**A**] describe stability: stable for *λ* < 0, unstable otherwise.

Oscillatory motion

For **complex conjugate** values of *λ* (which are quite common, see the plot of forward speed vs the imaginary component of the eigenvalues to the right), the solution (and hence the bike) will exhibit oscillatory motion (this can be predicted from Euler's equation: $e^{xi} = \cos(x) + \sin(x)i$ where $i = \sqrt{-1}$: an imaginary number), either growing or shrinking, depending on the sign of the real part of the eigenvalues.

The frequency of the oscillation is proportional to the magnitude of the complex number.

In general, during oscillation the steer and lean may be out of phase somewhat.

Superposition

The property that the sum of two solutions is also a solution.

Pick a part of speed range with one unstable eigenvalue. Then pick an initial condition, expressed as a linear combination of modes 1, 2, and 3 (plus phase). Each of those has its own lean-steer plot, and they can be added together to show the total lean-steer behavior.

Linearized Equations of Motion

For the idealized bike that JBike6 uses, there are only 2 coordinates required to describe the bike configuration at any time:

1. lean angle of the rear frame relative to ground plane,

2. steer angle of the front fork relative to the rear frame, and

Second-Order Ordinary Differential Equations (ODEs)

Two second-order ordinary differential equations represent the motion of the bike system. [3]:

$$M_{\theta\theta}\ddot{\theta}_r + K_{\theta\theta}\theta_r + M_{\theta\psi}\ddot{\psi} + C_{\theta\psi}\dot{\psi} + K_{\theta\psi}\psi = M_{\theta_r}$$ (the 'lean' equation)

$$M_{\psi\psi}\ddot{\psi} + C_{\psi\psi}\dot{\psi} + K_{\psi\psi}\psi + M_{\psi\theta}\ddot{\theta}_r + C_{\psi\theta}\dot{\theta}_r + K_{\psi\theta}\theta_r = M_{\psi}$$ (the 'steer' equation)

where[4]

θ_r is the lean angle of the rear assembly

ψ is the steer angle of the front assembly relative to the rear assembly

M_θ and M_ψ are moments (torques) applied at the rear assembly and the steering axis, respectively. For the analysis JBike6 performs of an uncontrolled bike, these are both taken to be zero.

Matrix Form

These equations of motion can be written in matrix form:

[3] Hand Masters Thesis, Cornell, 1988, page 23.
[4] page 35.

$$\left[\overbrace{\begin{pmatrix} M_{\theta\theta} & M_{\theta\psi} \\ M_{\psi\theta} & M_{\psi\psi} \end{pmatrix}}^{M} D^2 + \overbrace{\begin{pmatrix} C_{\theta\theta} & C_{\theta\psi} \\ C_{\psi\theta} & C_{\psi\psi} \end{pmatrix}}^{C} D + \overbrace{\begin{pmatrix} K_{\theta\theta} & K_{\theta\psi} \\ K_{\psi\theta} & K_{\psi\psi} \end{pmatrix}}^{K}\right] \begin{pmatrix} \theta_r \\ \psi \end{pmatrix} = \begin{pmatrix} M_{\theta_r} \\ M_{\psi} \end{pmatrix}$$

Where

M is the mass matrix,

C is the damping matrix,

K is the stiffness matrix, and

D is the differential operator.

Note that for the upright bike $C_{\theta\theta} = 0$, and so it does not appear in the second-order ordinary differential equations above.

JBike6 combines the *M*, *C*, and *K* matrices into 2 4x4 matrices, *A* and *B*,

$$B = \begin{bmatrix} M_{\theta\theta} & M_{\theta\psi} & 0 & 0 \\ M_{\psi\theta} & M_{\psi\psi} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } A = \begin{bmatrix} -C_{\theta\theta} & -C_{\theta\psi} & -K_{\theta\theta} & -K_{\theta\psi} \\ -C_{\psi\theta} & -C_{\psi\psi} & -K_{\psi\theta} & -K_{\psi\psi} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and calculates the 4 *generalized eigenvalues* of them via:

$Ax = \lambda Bx$

Where the values of λ that satisfy the equation are the generalized eigenvalues and the corresponding values of *x* are the generalized right eigenvectors. Read more about generalized eigenvalues in MATLAB's on-line help about the '**eig'** function.

Read more about working with these equations in the Appendix below.

Interpretation and Verification of Results

Interpretation

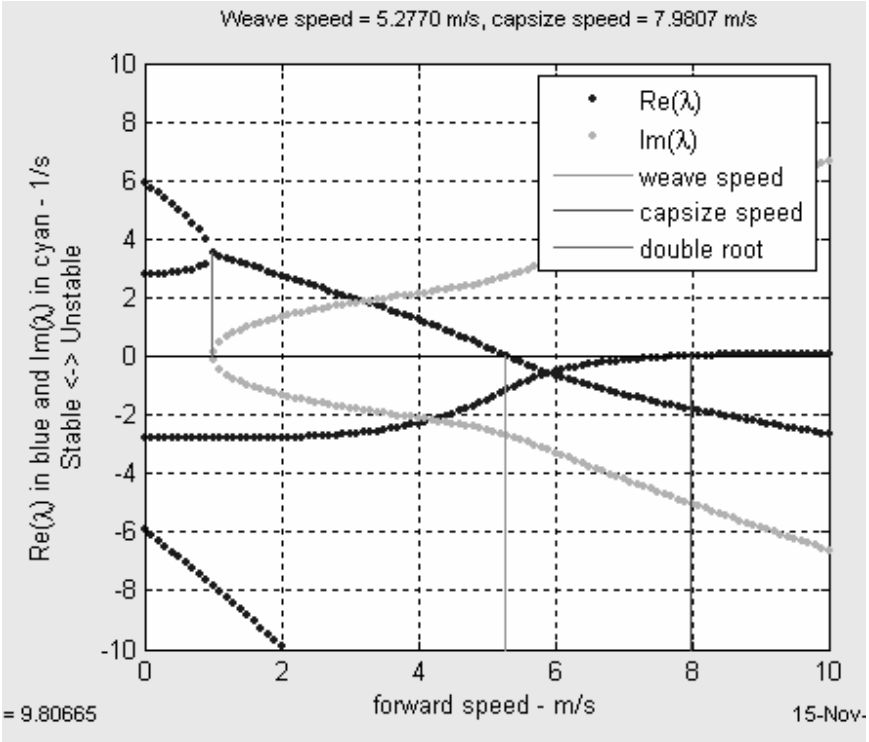
As an example consider, the eigenvalue plot for the Schwinn Crown.

At forward speeds of 0 - 1 m/s: up to emergence of complex conjugates

Large positive eigenvalues: very unstable. Two pos/neg pairs – both falling and uprighting. [Only one pair is seen for a simple inverted pendulum.] One pair corresponds to when the steering is turning toward the lean; the other when it is turning opposite to lean.

No imaginary parts of eigenvalues: not oscillating.

Just what you would expect from a bike standing still or nearly so. It will simply flop over.



At forward speeds of 1 - 5.2770 m/s: up to the weave speed

Positive eigenvalues, but decreasing in magnitude: unstable, but decreasingly so.

Eigenvalues with imaginary parts: oscillating with increasing frequency, and the rate of increase is rapid at first, but then slows.

The bike will weave back and forth one or more times before falling over.

At forward speeds of 5.2770 - 7.9807 m/s: the stable range: between the weave and capsize speeds

No positive eigenvalues, and increasingly negative: stable, and increasingly so.

Eigenvalues with imaginary parts: oscillating.

The bike will weave back and forth, less so each time, and eventually roll straight ahead, although not necessarily in the original direction.

At forward speeds of 7.9807 m/s and higher: above the stable range

Small positive eigenvalue: unstable

Eigenvalues with imaginary parts, but whose real component is much smaller than the positive eigenvalue: oscillations overwhelmed.

The bike slowly leans farther and farther to one side, without oscillation, until it finally falls over.

The transition between the behaviors in these speed ranges is as smooth and gradual as the curves in these plots suggest and can easily be seen in a numerical simulation with the same parameters.

That a bike loses stability at speeds higher than the capsize speed does not mean that it becomes harder to handle. This ‘capsize’ instability is very slow, on the order of seconds, and easy for a rider to compensate for. It is nothing like the ‘shimmy’ that some bikes exhibit at high speeds. That is an altogether different phenomenon usually based on frame flexibility (in bicycles) or tire flexibility (in motorcycles).

That JBike6 does not find a stable range of forward speeds for a particular bike configuration does not mean that in the real world it cannot be ridden or even that it would be hard to ride. Conversely, a stable range, no matter how big, does not mean that a bike will be easy to ride. Reality is more complicated than that. Two major factors that come into play on a real bike are that the rider is not really rigidly attached, and the rider may apply torques at the steering axis and between his mass and the frame. Other significant factors include friction between the wheels and the ground and at the steering axis.

A typical goal of analysis might be to adjust bike parameters and note how the ‘stable speed range’ is displaced or increased. Or, instead of

uncontrolled STABILITY [negative eigenvalues], one might pay greater attention to BOUNDED GROWTH – a bike should be fairly rideable as long as no real part (or imaginary part with positive growth) exceeds about 4.0 in magnitude. In other words, one concept of a reasonable controllability criterion is that any eigenvalues in the right half plane should remain within a semicircle of radius 4.0.

Verification

The equations of motion for an idealized, uncontrolled bike used in JBike6 have been independently verified in a paper by A. L. Schwab, J. P. Meijaard and J. M. Papadopoulos.

"The entries of the matrices of these equations form the basis for comparison. Three different kinds of methods to obtain the results are compared: pencil-and-paper [the equations used in JBike6], the numeric multibody dynamics program SPACAR, and the symbolic software system AutoSim. Because the results of the three methods are the same within the machine round-off error, we assume that the results are correct and can be used as a bicycle dynamics benchmark."[5]

Accelerator keys

CTRL+A	Draw axis lines on eigenvalue plot
CTRL+B	Draw coordinate axes on bike diagram
CTRL+D	Calculate and Draw Bike
CTRL+E	Draw additional eigenvalue plots in separate window
CTRL+F	Set 'save bike' directory…
CTRL+G	Draw grid lines on eigenvalue plot
CTRL+H	Using JBike6…
CTRL+I	Draw imaginary parts of eigenvalues
CTRL+J	Interpolate double roots from calculated eigenvalues
CTRL+K	Set gravity constant…
CTRL+L	Draw legend on eigenvalue plot
CTRL+M	Use Styles menu styles in additional plots
CTRL+N	Save eigenvalues…
CTRL+O	About JBike6…
CTRL+P	Print Bike…
CTRL+Q	Quick Reference Guide…
CTRL+R	Draw double roots with vertical lines
CTRL+S	Save bike parameters
CTRL+T	Display status messages during calculations

[5] Benchmark Results on the Linearized Equations of Motion of an Uncontrolled Bicycle, 2005, A. L. Schwab, J. P. Meijaard, and J. M. Papadopoulos, page 2.

CTRL+U	Print Setup…
CTRL+V	Draw weave speed with vertical line
CTRL+X	Exit
CTRL+Y	Save settings and styles when exiting
CTRL+Z	Draw capsize speed with vertical line

Note that changes in the ‘Styles’ menu do not take effect until you calculate and draw the eigenvalues again.

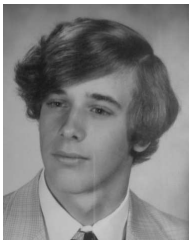
Appendices

Origin of JBike6

JBike6 is the successor of JBike5, a Fortran program which was written by William Krauss and Jim Papadopoulos around June 1988. Jim had worked closely with Scott Hand on his Master's Thesis at Cornell University: Comparisons and Stability Analysis of Linearized Equations of Motion for a Basic Bicycle Model

JBike6 was created by Arend L. Schwab of Delft University of Technology and Jim Papadopoulos of Cornell University around June 2003.

Professor Andy Ruina of Cornell University brought the team together.

			
Arend L. Schwab Assistant Professor Applied Mechanics Delft University of Technology	Jim Papadopoulos Contributing Author Bicycling Science : Third Edition and Star of Junkyard Wars	Andy Ruina Professor Theoretical & Applied Mechanics Cornell University	Andrew Dressel Graduate Student Theoretical & Applied Mechanics Cornell University

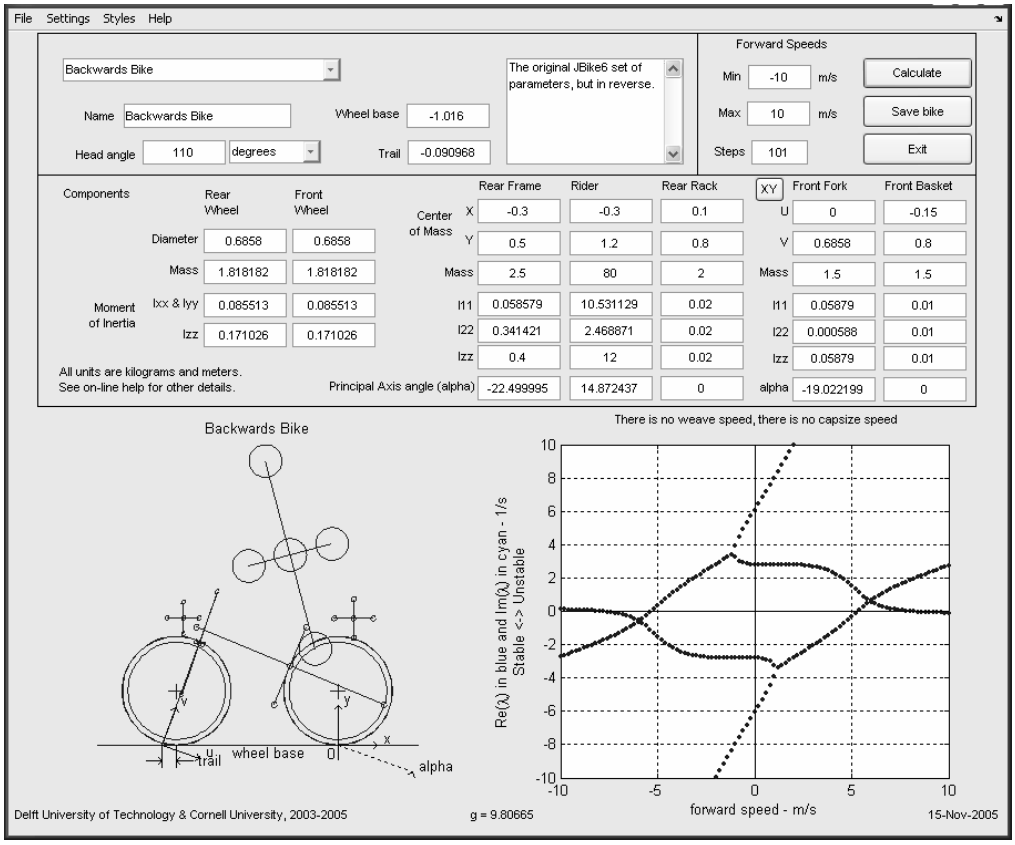
Rear steering

JBike6 provides a couple of options for examining the behavior of a bike with rear steering.

1. Simply specify a negative speed. This reveals that the eigenvalues are symmetrical about the x-axis exactly at the Re(λ)-axis, and that they have a wonderful inverse or mirror symmetry about the Re(λ)-axis.

2. Specify a backwards bike with negative wheelbase, trail, a head angle greater than 90°, etc.

There is not a hard line defined between front steering and rear steering. Instead there is just a continuous spectrum from most of the mass in the rear to most of the mass in the front, head angle positive to head angle negative, and trail positive to trial negative.



More on Gravity

The standard sea-level value of g in physics tables is 9.80665 m/s^2. It was set at the by the 3rd Conférence générale des poids et mesures (General Conference on Weights and Measures) in Paris in 1901. Read more at Wikipedia.

Little '*g*' described above incorporates large '*G*', the universal gravitational constant that expresses the gravitational attraction between two bodies, the radius of the earth, the mass of the earth, variations in the earth's density, and the so-called 'centrifugal force' due

to the rotation of the earth. Little '*g*' varies slightly from location to location on the surface of the earth.

The Physics Department at Montana State University, Bozeman has posted a table of local gravitational acceleration constants from around the world. They vary from 9.7724 in Quito, Ecuador to 9.8189 in Helsinki, Finland.

More on Precision

MATLAB, the underlying mathematics software of JBike6, assigns numbers the data type double, which means that they are double-precision floating-point numbers that are 8 bytes (64 bits) in length. The internal representation is always the 8-byte floating-point representation provided by the particular computer hardware. On computers that use IEEE floating-point standard arithmetic (ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic) (Intel Pentium and Celeron processors, for example), the resulting internal value is truncated to 53 bits. This is approximately 16 decimal digits.

The smallest magnitude is 4.9407e-324, and the largest is 1.7976e+308

You can see an example of the side effects due to the necessary conversion to base 2 and the finite nature of Standard 754 with these four operations in MATLAB:

```
a = 4 / 3
b = a - 1
c = 3 * b
e = 1 – c
```

They yield e = 2.220446049250313e-016, which is not the expected 0.0 from the 16th digit on.

You can read more about the IEEE standard 754 and MATLAB in 'Numerical Computing with MATLAB' by Cleve Moler.

More on Parameters

Wheelbase = X of front wheel – X of rear wheel. JBike6 needs some nonzero fore-aft extent for things to make sense. Once again there is a dimensional scaling such that a shrunk bike behaves just like a full-sized bike, except over a shrunken speed range. And once again, while the limit of 'zero' for all lengths could be approached, it will only make sense if ratios are preserved. So you should think of fixing one dimension and varying the others.

Trail = X of steer axis – X of front wheel

Head angle: JBike6 produces identical results for values ±360, when using the uv-coordinate system, and ±180 when using the xy-coordinate system.

Wheel radius is irrelevant as long as the contact point is fixed and the ratio between inertia and radius (I/r) is fixed, while wheel-plus-assembly mass and cm are unchanged. JBike6 could actually use a much reduced input set, but uses traditional parameters for convenience and compatibility with other calculations.

Wheel Inertia: For a conventional wheel, $Izz < mr^2$ and $Izz < Ixx + Iyy$ must both be true. However, for a wheel geared to a slower-moving or faster-moving flywheel, which is located off the ground, these restrictions do not apply to Izz.

Mass: The eigenvalues remain unchanged by mass scaling, so there is nothing to learn by driving all masses to zero. You might as well fix one mass in the system and adjust all the others.

Inertia: JBike6 calculates the same result if you add or subtract 180° to the inertial principal axis angle or if you add 90° t o the inertia principal axis angle and switch the values in I_{11} and I_{22}. I_{11} does not need to be the larger value. Read more on Inertia Tensors at Wolfram Research and Wikipedia.

Conventionally, the **y coordinate** of every mass center should be positive. An exception might be a bike on an elevated path or walkway, with a ‘counterweight’ putting its cm below ground level.

Conventionally, a **rider center of mass** can not exist within the wheel radius, but it is possible with heavy saddlebags.

Conventionally, the overall center of mass should fall within the wheelbase or the bike will pitch over, unless wheels are magnetic or otherwise held to the ground.

More on Eigenvalues

“Eigenvalues are a special set of scalars associated with a linear system of equations (i.e., a matrix equation) that are sometimes also known as characteristic roots, proper values, or latent roots (Marcus and Minc 1988, p. 144).” Read further at MathWorld by Wolfram Research.

Here is another way to think of eigenvalues. First, consider an eigenfunction, the essence of which is 'solutions which evolve to remain self-similar'. Anything which evolves in such a fashion only changes magnitude, in proportion to current magnitude, i.e. exponential behavior. In a sense the determination of eigenfunctions is the key question, and the resulting eigenvalue can then be determined by plugging in the eigenfunction and noting the magnitude change [after one multiplication of a transition matrix, which could be based on an elapsed time interval].

In fact, you may consider the classical approach, given initially above, starting with eigenvalues, to be backwards. It is merely the trick that most quickly leads to finding the eigenfunctions, or indeed the eigenvalues after recognizing that the eigenvalues alone are needed to answer a stability question.

For the upright bike, any perturbation motion consists of time evolving lean and steer, with initial conditions on lean amount, lean rate, steer amount, and steer rate. The whole theory of eigenfunctions is that there is a complete set of special 'fundamental' motions (which evolve in self-similar fashion), and since the system is linear, any disturbance can be decomposed into fundamental motions, and its whole future evolution easily calculated when they are added together.

Any linear time-evolving, time-independent equations have the property that the sum of two solutions is also a solution. Therefore there is special value in looking for 'eigensolutions' - solutions that remain self-similar as they evolve, and hence evolve exponentially. Then any solution can be expressed as linear combinations of eigensolutions. For the purpose of studying stability, only the exponential growth, defined by the 'eigenvalues', is really needed.

So, eigenvalues are often found by relatively simple calculations, on the way to determining eigensolutions. For example when written in state space form, eigenvalues they are identified as 'expansion numbers' that make it possible to find self-similar (exponential) solutions, by the vanishing of a determinant (polynomial).

You can read more at Numerical Computing with MATLAB at The MathWorks

Another excellent on-line article about eigenvalues, eigenvectors, eigenspaces, and some of their applications can be found in Wikipedia.

$$[I_{cm}] = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$$

One example of the utility of eigenvalues and their corresponding eigenvectors can be seen in inertia tensors. In general, it takes 6 numbers (I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, and I_{yz}) to describe the inertia of a 3-dimensional object about its center of mass (*cm*). These are often arranged in a 3x3 matrix where 3 of the numbers are repeated (I_{xy}, I_{xz}, and I_{yz}) to symmetrically fill out the 9 locations. It turns out, though, that if the coordinate system used, with the origin always at the center of gravity, is oriented properly, then only the 3 numbers along the diagonal of the matrix (I_{xx}, I_{yy}, I_{zz}) are non-zero. In fact, these three numbers are the eigenvalues of the 3x3 matrix, and the 3 axes of the necessary coordinate system are the eigenvectors. This is how JBike6 only needs three numbers (I_{xx}, I_{yy}, I_{zz}) to describe the inertia of the parts of a bike when in general 6 numbers are required. It also needs a Principal Axis Angle, the amount that the principal axis is rotated from the horizontal, to complete the description

For more on how to generate eigenvalues and eigenvectors, see the next section.

More on Inertia Tensors

JBike6 provides only 3 spaces to specify the inertia tensors for the parts of a bike, whereas, in general, a 3-dimensional inertia tensor has 9 values usually arranged in a 3-by-3 symmetric matrix. In order to generate the 3 values that JBike6 requires, you can diagonalize the 3-by-3 matrix with MATLAB's eig() function. The output of eig() is two matrices: the second is a diagonalized matrix with the desired 3 values (actually eigenvalues) along its diagonal, the first is a matrix with eigenvectors for columns

```
[V,D] = EIG(X);
I11 = D(1,1);
I22 = D(2,2);
Izz = D(3,3);
```

These eigenvectors are the axis necessary to express the inertia tensor with just 3 values. Because the eigenvectors are orthogonal and JBike6 requires symmetry about the x-y plane, you can simply use MATLAB's atan() function to calculate the principal axis angle (alpha) that JBike6 needs from either the first or the second column of the second matrix returned by the eig() function:

```
Principal_axis_angle_in_deg =
atan2(V(2,1),V(1,1))*180/pi();
```

For further discussion about eigenvalues, eigenvectors, and inertia tensors, see the last paragraph of the previous section.

More on the Bike Diagram

Any rigid body inertia tensor has three mutually perpendicular principal axes. It is possible to reproduce the inertia of a body by placing point masses out at the ends of three orthogonal rods. For the graphical representation, JBike6 divides a body's mass into six equal lumps, each with 1/6 of the total mass. Then the rod half-lengths L_1, L2 and Lz match the inertia tensor, deduced from the formulas

$$I_{11} = 2(m/6) * [L_2^2 + L_z^2]$$
$$I_{22} = 2(m/6) * [L_1^2 + L_z^2]$$
$$I_{zz} = 2(m/6) * [L_1^2 + L_2^2]$$

Solved as $(I_{11} + I_{22} – I_{zz}) = 2(m/6) * [2L_z^2]$, or $L_z^2 = (I_{11} + I_{22} – I_{zz}) / (4m/6)$, etc.

Thus the mass moment of inertia about any one axis is large, if the two perpendicular rod lengths are large.

JBike6 indicates the size of the point masses by presenting them as modest size uniform-thickness disks, whose area indicates their mass. This is just a graphic convention; the mass-centric inertia of each disk is not considered when computing the overall moment of inertia. These are point masses, or masses pivoted at their centers so they don't rotate, only translate.

Consider the disks to be made of steel [density taken as 7874 kg/m^3], and with a thickness of two inches or 0.0508 m, radius R is defined from

$$M/6 = (\text{density}) * (\text{volume}) = (7874) * (\pi R^2 * 0.0508) = 400 * \pi R^2$$

The radial thickness of a wheel is shown in a consistent way – treating the rim as made of 2 inch thick steel, we write $400\pi(R_0^2 – R_i^2)$ = Mass of wheel, which permits us to solve for R_i.

More on the Equations of Motion

As JBike6 calculates the eigenvalues, it displays in the MATLAB command window, if you have the 'display status messages during calculations' option checked in the 'Settings' menu, the precise values of the equation coefficients of the M, C, and K (the sum of K0 and K2) matrices. You can add the necessary speed dependence simply by multiplying C by the forward speed and K2 by the forward speed squared. In MATLAB:

```
M = M0;
C = C1 .* v;
K = K0 + K2 .* v^2;
```

With these equation coefficients, you can integrate to get the resulting bike motion.

Three cases are of special interest:

A. If the steer angle, as a function of lean angle from a control algorithm for example, is given, take the lean equation and put all steer terms on the right hand side. Then integrate to calculate bike lean.

B. If the steer torque, from a controller based on lean and steer angles for example, is given, you need the steer equation too. Then integrate for lean and steer angle.

C. If there is no steer control or torque, a special case of B, integrate both equations with moments equal to zero.

You can read about Integrating Ordinary Differential Equations (ODEs) at Mathworks.

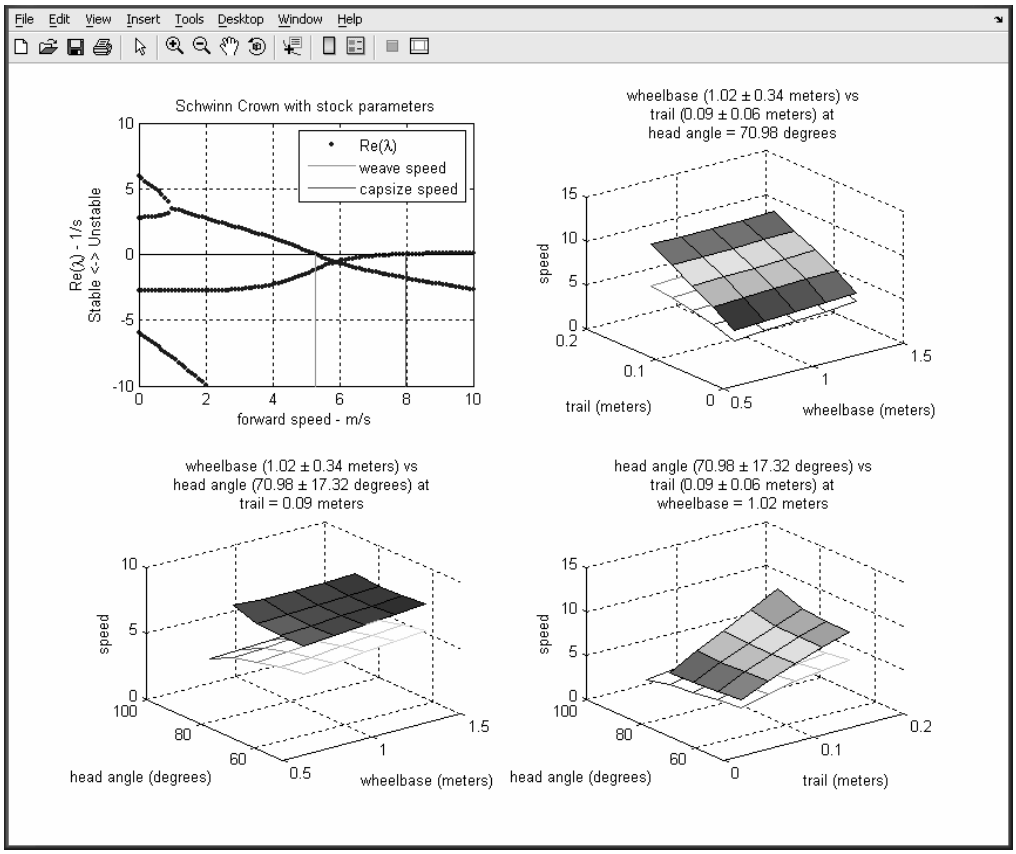
More on Running JBike6 without the GUI

The included MATLAB file, JBike6Batch.m, gives another example of running JBike6 without the GUI. By calling the appropriate JBike6 modules, JBike6Batch.m calculates the weave and capsize speeds for a range of wheelbase, head angle, and trail values for a give bike. It then plots surfaces of these critical speed values against the parameters that generate them.

You can run JBike6Batch.m in MATLAB with any of the methods described above for running JBike6.m: from the MATLAB command-line or from the MATLAB editor.

By editing this MATLAB source file, you may specify the bike to analyze, the minimum values of wheelbase, head angle, and trial, and the number of different values of each to consider. JBike6Batch calculates all the required parameter values, the resulting critical speeds, and plots the surfaces as can be seen below. The top, solid surface is the capsize speed, and the lower, mesh surface is the weave speed.

As with the 4 additional eigenvalue plots, you can use all of MATLAB's plot manipulation tools including zoom, pan, and 3D rotate.



In this way, you can gain some idea of how varying key bike parameters will affect its hands-free stability.

Component Files

JBike6.m	Main module, calls JBike6GUI.m
JBike6GUI.m JBike6GUI.fig	Graphical user interface, calls JBConfig, JBini, JBfig, JBmck, JBvcrit, JBeig, and JBref
JBconfig.m	Contains user savable parameters
JBini.m	Contains default parameters
JBini_files	Subdirectory with saved bike JBini*.m files
JBfig.m	Draws representation of bike in main window
JBdrawcross.m	Draw a cross to represent wheel hubs
JBdrawinertia.m	Draw the inertia 'dumb bells'
JBdrawcircle.m	Draw circles
JBmck.m	Calculates the linearized equations of motion
JBaddb.m	Adds the mass and inertia of rigid bodies into one body
JBvcrit.m	Calculates the weave and capsize speed
JBcombint.m	Combines two interval lists A and B into a new interval list C
JBpolyint.m	Makes an interval list A in x where the polynomial p(x) is positive.
JBdoubleroots.m	Calculates double root speed(s) analytically (uses Symbolic tool box)
JBroots.m	Calculates double root speed(s) by interpolation from eigenvalues
JBeig.m	Calculates the eigenvalues and eigenvectors
JBrev.m	Plots eigenvalues in main JBike6 window
JBplot4.m	Plots eigenvalues four ways in separate window
JBhelp.htm	Main help file
JBhelp_files	Subdirectory of support files: images
JBQRef.htm	Main quick reference file
JBQRef_files	Subdirectory of support files: images
Readme.txt	A few notes about how to get started with JBike6
License.txt	Complete text of license agreement
JBike6Batch.m	A sample file demonstrating how to call the individual JBike6 components directly
JBike6Control.m	A sample file demonstrating how to plug JBike6 into MATLAB's Control System Toolbox

See the next section on JBike6 Architecture for more details

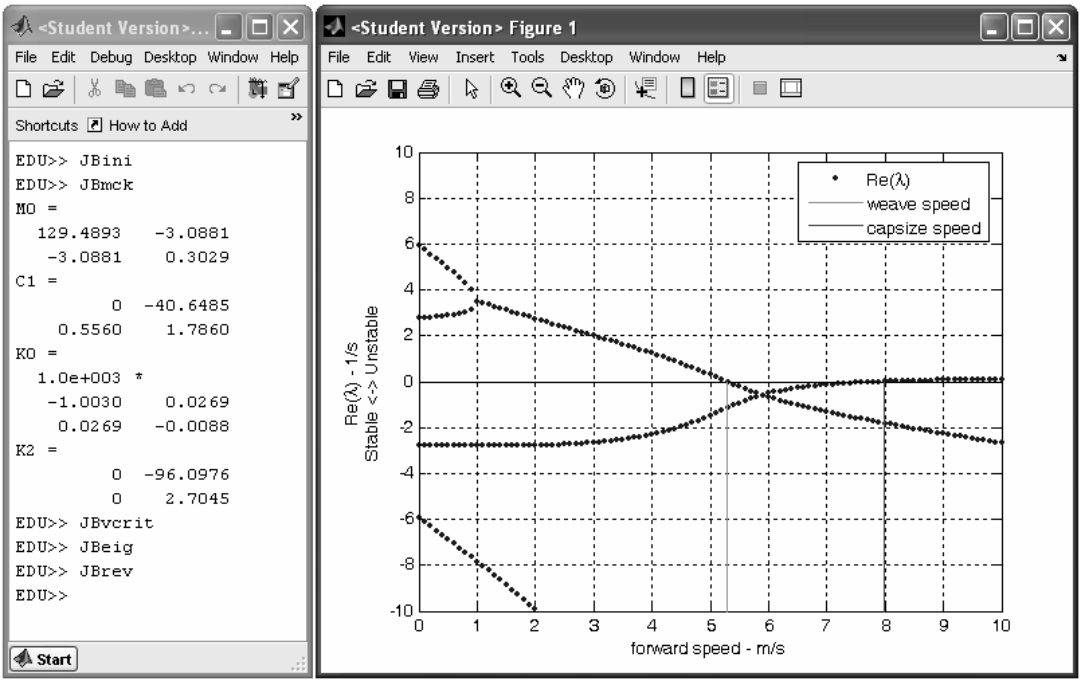
JBike6 Architecture

The architecture of JBike6 and its graphical user interface is designed to provide both ease-of-use and versatility.

Ease-of-use is provided by JBike6GUI.m and its companion file JBike6GUI.fig. These are created with MATLAB's guide utility, and they create the entire user interface. By necessity, parameters are stored in MATLAB's handle structure with the guidata() function so that they can be available to various call-back functions.

Versatility is provided the simple and flat structure of the mathematical modules. They contain MATLAB scripts that do not even need to pass parameters to each other. They simply define variables in MATLAB's global space and access them as necessary. You can run them directly from the MATLAB command line and manipulate the variables as you go.

For example, you can load any set of bike parameters you want simply by running the appropriate JBini*.m file. Then, to calculate the linearized equations of motion, simply run JBmck.m. You can calculate the critical speeds with JBvcrit.m, the eigenvalues with JBeig.m, and then plot them with JBref.m. The calculated values of the critical speeds are contained in the vweave and vcapsize variables respectively.



See the previous section on the component files of JBike6 for more details.

Links

Bicycle Mechanics at Cornell University's The Biorobotics and Locomotion Lab

JBike6 home page

MATLAB help by The MathWorks

MathWorld by Wolfram Research.

Numerical Computing with MATLAB by Cleve Moler

Wikipedia.

License

Use of JBike6 is licensed to you by the authors and limited by the license agreement. You agreed to its terms and conditions when you downloaded JBike6. A copy of the license agreement is included with the JBike6 program files in license.txt. You can also find a copy on the JBike6 website.

Glossary

12-coordinate system

Read as 'one-two coordinate system'. The coordinate system associated with each rigid body with its origin at the center of mass, and rotated, in this case about the z-axis, so that only 3 numbers, instead of the usual 6, are required to completely specify its inertia properties in 3D. In JBike6, you can specify the angle, called the principal axis angle, or alpha, that the 12-coordinate system is rotated from xy. The 1-axis is rotated counter-clock-wise from the x-axis, the horizontal, by alpha, and the 2-axis is rotated in the same direction from the y-axis, the vertical, by the same amount.

Alpha

The principal axis angle. The amount, for each body, that its 12-coordinate system (read as one-two, not twelve) is rotated about the z-axis from the xy-coordinate system.

ASCII text

American Standard Code for Information Interchange. A standard for assigning numerical values (7 binary digits) to the set of letters in the Roman alphabet and typographic characters. Read more at Wikipedia.

Bicycle

In this document, **bicycle** is the term used to represent a bike powered only by a rider, as opposed to a motorcycle. A bicycle is typically lighter than a motorcycle, especially relative to the rider, and usually has narrower tires.

Bike

In this document, **bike** is the term used to represent a device with two wheels connected to a frame and fork, one behind the other, and with a pivot so that one wheel may be turned with respect to the other for steering: either a bicycle or a motorcycle.

Capsize speed

The forward speed at which capsize motion neither grows nor decays. This is represented by a vertical line on the eigenvalue plot.

Capsize motion is that motion in which the bike slowly leans and steers, without oscillation, farther and farther until it falls over.

At speeds above this capsize speed, the capsize motion will progress until the bike falls over. There may not be a capsize speed for a particular bike configuration. Odd designs have been found which have no finite capsize speed, and are stable at all speeds above the weave speed.

Center of mass

The point in space, not necessarily in or on the body, at which all the mass of the body may be considered to be concentrated for many mechanics calculations.

Command-line Interface

A means of interacting with an application through typed text strings, commands, as opposed to a GUI.

Complex number

A mixture of real and imaginary numbers, usually written *a*±*bi*, where *a* is the real component, *b* is the imaginary component, and $i = \sqrt{-1}$. Complex numbers often arise as roots of equations.

Complex conjugate

The complex number associated with another complex number but with the opposite sign between the real and imaginary parts. The complex conjugate of *a*-*bi* is *a*+*bi*. Complex roots of equations usual come as complex conjugate pairs.

Coordinate system

An origin and set of orthogonal axes, oriented according to the right-hand rule if in 3 dimension, that can be used to specify locations. JBike6 uses 3 different coordinate systems to completely specify a bike and its rider: xy, uv, and 12, all of which share a common z-axis.

Coupled equations

Equations that share one or more variables so that they depend on each other: they cannot be solved independently. The equations of motion for a bike are a set of 2 coupled second-order ordinary differential equations.

Damped harmonic oscillator equation

A common equation from classical mechanics that describes the motion of a damped harmonic oscillator: a mass, spring, dashpot. It is often written $m\ddot{x} + c\dot{x} + kx = 0$ where *x* is a coordinate describing the location of an object, $\dot{x}$ is velocity of the object: the first derivative with respect to time of coordinate *x*, $\ddot{x}$ is the acceleration of the object: the second derivate with respect to time of the coordinate *x*, *m* is the mass of the object, *c* is the damping coefficient, and *k* is the spring constant. The damping coefficient, *c*, describes how rapidly the dashpot damps the system, and the spring constant, *k*, describes how stiff the spring is.

Damping matrix

A matrix of the linearized equations of motion for a bike that multiplies the velocities, as does the damping term in classical damped harmonic oscillator equation.

Degrees

Angular unit of measure: equivalent to 0.0174533 radians. There are exactly 360° in a complete circle: 2 π radians.

Degrees of freedom

The number of coordinates necessary to completely describe location and orientation. In 3-space, a 3-demensional body requires 6 coordinates (x, y, z, roll, pitch, and yaw) and so has 6 degrees of freedom. If the body is in motion, then it has 6 more degrees of freedom, the rate at which the first 6 change. For the analysis that JBike6 performs, the bike only has 2 degrees of freedom: lean angle of the rear frame, and steer angle of the front fork.

Differential equations

Equations of one or more variables and their derivatives.

Differential operator

A symbol that indicates that a variable is differentiated, often with respect to time. A superscript number to the right indicates how many differentiations.

Double roots

In JBike6, the forward speeds at which real eigenvalues come together and then split apart as complex conjugates. This signals the onset of oscillation. There may be more than one, and JBike6 indicates them by vertical lines on the eigenvalue plot.

Dumbbells

A technique JBike6 uses to graphically represent the rotational inertia properties of a body in the bike diagram. The mass of the body is distributed into 6 equal disks, drawn as circles and only 5 of which are visible in 2 dimensions, and placed on the end of 6 arms, only 4 of which appear in 2 dimensions. The arms are placed along the axes of the inertia tensor for the body. The

length of the arms represents the resistance of the body to angular acceleration about the other two axes.

Eigenvalue Plot

In JBike6, a plot of the forward speed vs the corresponding eigenvalues of a particular bike configuration.

Eigenvalues

Eigenvalues are extensively covered above in two sections: Eigenvalues and More on Eigenvalues.

Equations of Motion

Equations that tell how a body, or system of bodies, move in time: usual second-order differential equations. In JBike6, the equations of motion for a bike are two, coupled, second-order differential equations.

Equilibrium

A point at which a system is in balance: does not move. Equilibrium points may be stable or unstable. For a pendulum with a rigid arm, straight down is a stable equilibrium point, and straight up is an unstable equilibrium point: any perturbation will cause the pendulum to move exponentially away from that straight up position.

Euler's equation

One of many amazing discoveries made by 18th century Swiss mathematician Leonhard Euler: $e^{xi} = \cos(x) + \sin(x)i$, where x is any real number, i is an imaginary number, and e is the base of natural logarithms. See further explanation and a proof at Drexel's Math Forum.

Exponential growth or decay

A description of the motion of a system, especially with respect to equilibrium. Exponential growth is motion away from equilibrium, and exponential decay is motion towards equilibrium. In either case, the motion may be oscillatory. The expression 'exponential' is due to a commonly assumed form of the solution to the equation(s) of motion: $x = e^{\lambda t}$ where x represents the position of the system, t is time, e is the base of the natural logarithms, and λ is a constant. If λ is positive, then x

will grow for increasing values of *t*. If *λ* is negative, then *x* will decay for increasing values of *t*. If *λ* is zero, then *x* will remain constant: be in equilibrium. If *λ* is complex, then the motion will be oscillatory.

Fork

Usually the front part of a bike, that connects to the front wheel and the frame with revolute joints.

Forward speeds

Forward speed is a variable in the equations of motion for a bike, and so the eigenvalues depend on forward speed. In JBike6, the units are meters-per-second, and you can specify a range of forward speeds to investigate. The word ‘speed’ is used, instead of ‘velocity’, a vector quantity, because the direction is not defined, other than that it is forwards instead of backwards.

Frame

Usually the rear part of a bike, where the rider sits, that connects to the rear wheel and the fork with revolute joints.

Frequency

The rate at which a motion repeats: measured in ‘cycles per second’ or ‘hertz’. Used to measure oscillatory motion.

Graphical User Interface (GUI)

Pronounced "gooey". A means of interacting with an application through a metaphor of direct manipulation of graphical objects, as opposed to a ‘command-line’ interface.

Gravitational constant

The gravitational constant extensively covered above in two sections: Gravity and More on Gravity.

Head angle

The angle that the bike head tube (steering axis) makes with the horizontal. By convention, positive angles are clockwise from the horizontal. A 90° head angle would be vertical. Comm on head angles for road bikes range from 70° to 75°.

Head tube

The part of a bike frame in which the fork pivots.

Imaginary component

The part of a complex number which is an imaginary number.

Imaginary number

A number which is a real number multiple of $i = \sqrt{-1}$.

Inertia

The property of matter that resists acceleration, either linear or rotational.

Inertia tensor

A description of a body's inertia about its center of mass. Usually express in a 3x3 matrix of 9 values, 3 of which are repeated. If the 3 axes are oriented properly, then only 3 values are necessary.

Kilograms

Unit of mass used by JBike6: equivalent to 0.0685218 slugs. A kilogram weighs about 2.2 pounds at the surface of the earth.

Knife-edge wheels

An idealization use to simplify the equations of motion in JBike6's bike model. It ignores the kinematics of leaning a bike with finite-radius tires.

Lean Angle

The angle that the rear of a bike makes with the vertical. When the bike is straight upright, the lean angle is zero.

Linear Equations

Equations that contain only 'linear' terms. Linear terms satisfy the following two properties: Additivity: f(x + y) = f(x) + f(y) and Homogeneity: f(αx) = αf(x) for all α. Examples of nonlinear terms include x^2 and sin(x). Read more about linear and nonlinear systems.

Linearized Equations

Equations that have been simplified to be linear. A common 'linearization' is that for small angles θ, sin(θ) is approximately θ. The equation of motion for a simple pendulum, $m\ddot{\theta} + {}^{g}\!/_{\ell} \sin(\theta) = 0$, may be linearized in this way to $m\ddot{\theta} + {}^{g}\!/_{\ell} \theta = 0$ which then has the closed-form solution $\theta = C_1 e^{(g/\ell)t} + C_2 e^{(-g/\ell)t}$.

Mass

The property of matter that gives it inertia and gravitational attraction.

Mass matrix

A matrix of the linearized equations of motion for a bike that multiplies the accelerations, as does the mass term in classical damped harmonic oscillator equation. In JBike6 it consists of the mass and the inertia values of the particular bike configuration.

MATLAB

"MATLAB is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages." Read more at The MathWorks web site.

Matrix

A regular grid of values. Systems of linear equations and inertia tensors can both be expressed as matrices.

Meter

Unit of length used by JBike6: equivalent to 3.2808399 feet, 39.3700787 inches, 100 centimeters, 1000 millimeters, and 1.9102197 Royal Egyptian cubits.

Meters per second

Unit of speed used by JBike6: equivalent to 3.6 kilometers per hour, 2.2369363 miles per hour, mach 0.0029387, and 6,012.8727971 furlongs per fortnight.

Motorcycle

In this document, **motorcycle** is the term used to represent a bike powered by some type of engine or motor, as opposed to a bicycle. A motorcycle is typically heavier than a bicycle, especially relative to the rider, and usually has wider tires.

Ordinary Differential Equation (ODE)

Differential Equations distinguished from Partial Differential Equations (PDEs) which contain partial derivatives. The bike equations of motion that JBike6 uses are ODEs.

Origin

The location, part of a coordinate system, from which coordinates are measured.

Orthogonal

Perpendicular to. At right-angles to. The x, y, and z axes that JBike6 uses are mutually orthogonal.

Oscillatory motion

Motion which repeats itself. The swinging of a pendulum is oscillatory. For a bike, it is that motion in which the bike leans and steers back and forth.

Perturbation

A small change in the state of a body or system of bodies: often away from a point of equilibrium.

Precision

The number of digits used to perform a given computation. See the More on Precision section above.

Principal axis angle

In JBike6, the angle that 12-coordinate system (read as one-two, not twelve), used to express the rotational inertia of a body, is rotated from the xy-coordinate system.

Radians

Angular unit of measure: equivalent to 57.2957795 degrees. There are exactly 2π radians in a complete circle: 360°.

Real number

A number that is rational or irrational, not imaginary. Read more at Wikipedia.

Rear steering

A system in which the rear wheel pivots to steer a bike instead of the front wheel. There is not a hard line defined between front steering and rear steering. Instead there is just a continuous spectrum from most of the mass in the rear to most of the mass in the front, head angle positive to head angle negative, and trail positive to trial negative.

Revolute joint

A connection between two bodies that constrains them to pivot with respect to each other about a common axis. Door hinges are common physical examples of revolute joints. In the bike model used by JBike6, the steering axis and the wheel hubs are revolute joints.

Right-hand rule

A convention used to keep track of the orientation of axes in 3 dimensions. Start with the fingers of the right hand oriented in the direction of the first axis (x); bend the fingers in the direction of the second axis (y); the extended thumb then points in the direction of the third axis (z).

Rigid body

An idealized entity that permits absolutely no internal motion: no flex, no twist, no vibration, etc. The concept of rigid bodies is useful for drastically simplifying the analysis of real-world objects.

Rolling without friction

A theoretical situation in which objects roll without any loss of energy due to friction. Ignores paradox of why they would roll at

all instead of sliding. Useful for drastically simplifying the analysis of real-world objects.

Roots of an equation

Those values for which the equation equals zero: where the curve of the equation crosses the x-axis. The roots of a quadratic equation, $ax^2 + bx + c = 0$, are given by $x = \frac{-b \pm \sqrt{4ac - b^2}}{2a}$.

If b^2 is greater than $4ac$ then the roots are imaginary: a complex conjugate pair.

Second-order differential equations

Differential equations involving variables and their second derivatives. The damped harmonic oscillator equation of motion is an example of a second-order differential equation.

Self-stable

In the case of a riderless bike, the property of returning to straight and upright (zero lean angle and zero steer angle) motion after a perturbation of either lean angle or steer angle.

Stable

The property of an object or system of objects in which it returns to a point of equilibrium after a perturbation.

State-space form

A way of expressing equations of motion in which the derivative of all the coordinates are found by multiplying all the coordinates by a matrix.

Steer Angle

The angle, about the steer axis, between the front (fork and front wheel) and rear (frame and rear wheel) parts of a bike.

Steer Axis

The imaginary line, usually through a head tube, about which both the rear frame and the front fork of bike rotate in order to steer.

Stiffness matrix

A matrix of the linearized equations of motion for a bike that multiplies the positions, as does the spring constant in classical damped harmonic oscillator equation.

Symmetrical

The property of being the same on both sides of an axis or plane. The idealized bike that JBike6 uses is symmetrical about vertical plane down its middle: the left side and right side are mirror images of each other.

Tooltip

A small box containing helpful text that appears near a control of a GUI when you hover the mouse cursor over it.

Trail

The horizontal distance from the front hub (or ground contact point) to the point where the steering axis intersects the ground. By convention, positive trail is when the front wheel contact point is behind, toward the rear wheel, the steering axis intersect point.

Unstable

The property of an object or system of objects in which it moves away from a point of equilibrium after a perturbation.

uv-coordinate system

In JBike6, an optional coordinate system for expressing the location of the center of mass of the front fork and the front basket. Its origin is at the intersection of the steering axis and the ground plane and it is rotated so that what was its vertical axis is aligned with the steering axis.

Weave speed

The forward speed at which oscillatory motion neither grows nor decays. JBike6 represents this by a vertical line on the eigenvalue plot.

At speeds below this weave speed, the oscillations will grow until the bike falls over. There may not be a weave speed for a particular bike configuration

A conventional bike has a single nonzero speed interval of stability, and the weave speed is the lower boundary of this interval. Thus a bike that slows just below the weave speed, becomes unstable by weaving. But, if the weave speed is above the capsize speed, there probably is no stable interval.

Wheelbase

The horizontal distance between the hub (or ground contact point) of the front wheel and the rear wheel of a bike.

xy-coordinate system

In JBike6, the coordinate system for expressing the location of the center of mass of the rear frame and the rear rack. Its origin is at the rear wheel ground-contact point, directly below the rear hub.

z-axis

In JBike6, the coordinate axis that is ‘out of the plane’, that is orthogonal to the x and y axes, the u and v axes, and the 1 and 2 axes.

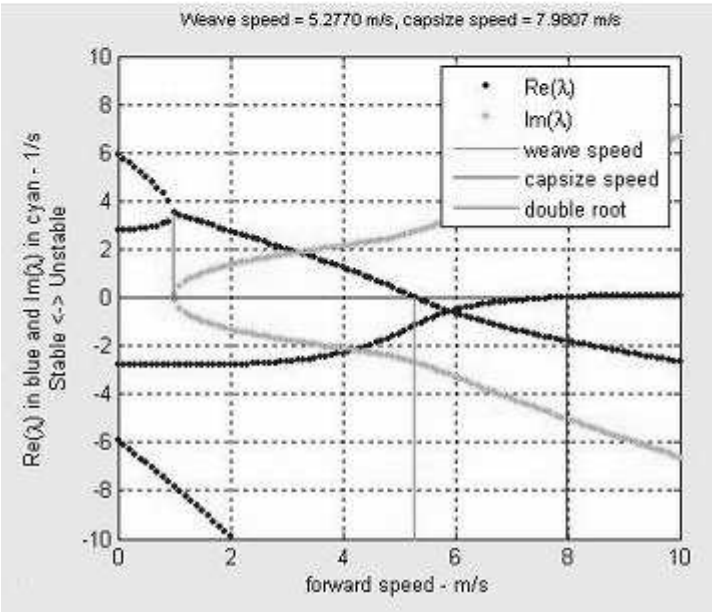
Complete Web Site

JBike6

A Benchmark for Bicycle Motion
(linearized ODEs)

JBike6 is a computer program that determines the stability of bicycles. You can enter information about the shape and mass distribution of a bicycle, and JBike6 calculates the speeds at which that bicycle is stable all by itself, with no control.

In particular, JBike6 calculates the eigenvalues (i.e., perturbation- growth exponents: λ in solutions of the form $q = ve^{\lambda t}$), for an idealized, uncontrolled bicycle. It then plots these linearized perturbation-growth eigenvalues over a range of forward speeds. For example, the bike shown at the right it is self-stable in the region of speeds (between 5.3 and 8 m/s) where both dark blue lines are below the x-axis.



The equations used in JBike6 are benchmarked exhaustively in a paper by A. L. Schwab, J. P. Meijaard, and J. M. Papadopoulos.

JBike6 has a graphical user interface (screen shot) and comprehensive on-line help.

If working on bicycle or motorcycle development or research, use JBike6 to:

- Calculate no-hands stability of a given design over the entire velocity range. The eigenvalue plot shows exactly at what speed, if any, a

configuration becomes stable, and at what speed, if any, it becomes unstable.
- Check the accuracy or validity of any other bicycle or motorcycle equations you may use. JBike6 provides several sets of numbers (benchmarked to high precision) that you can use for comparison.

JBike6 was created by:

- Arend L. Schwab, Assistant Professor of Applied Mechanics at Delft University of Technology. Wrote the main JBike6 engine in MATLAB.
- Jim Papadopoulos, Contributing Author of Bicycling Science : Third Edition. Created the JBike concept and monitored its development. Worked closely with Scott Hand on his Master's Thesis at Cornell University: Comparisons and Stability Analysis of Linearized Equations of Motion for a Basic Bicycle Model
- Andy Ruina, Professor of Theoretical & Applied Mechanics at Cornell University. Lab advisor.
- Andrew Dressel, Graduate Student of Theoretical & Applied Mechanics at Cornell University. Created JBike6 GUI, on-line help, and this JBike6 website.

Download a copy for free. Requires MATLAB, by The MathWorks, version 6.0 or higher.

Please email us with questions or comments.

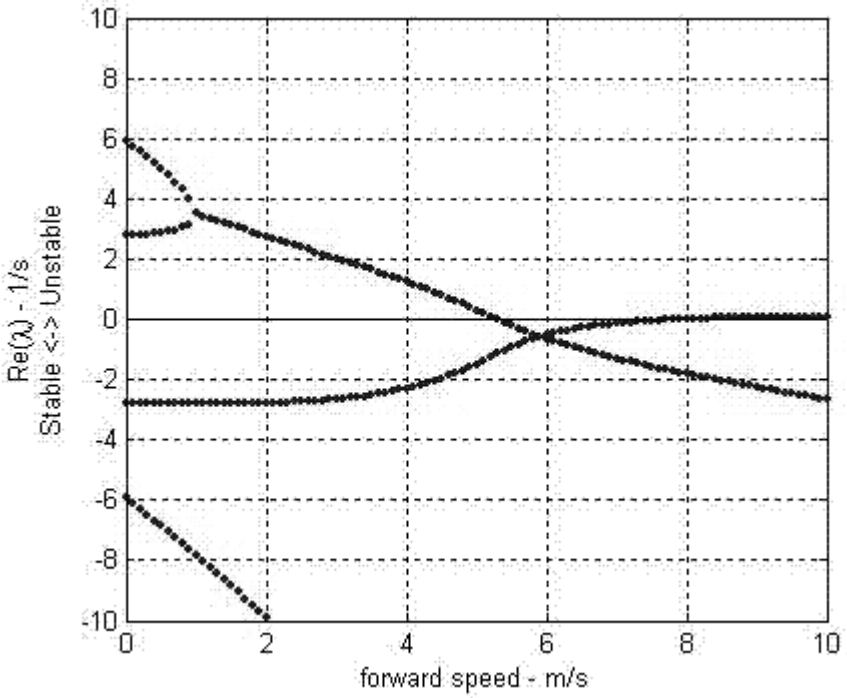
Last updated March 1, 2006

Copyright © 2003-2006 Schwab, Papadopoulos, Ruina, & Dressel, Delft University of Technology & Cornell University

JBike6 Self-Stability

Self-Stability

In general, a bike design is **self-stable** at those forward speeds for which the real parts of all the eigenvalues are less than zero. For example, the bike whose eigenvalues are shown at the right it is self-stable in the region of speeds (between 5.3 and 8 m/s) where both dark blue lines are below the x-axis.



Self-stable means that without external input, the bike will eventually roll straight and upright even if perturbed.) Such behavior can easily be seen in real-world bicycles, when rolled without a rider.

The bike must meet a couple simple criteria:

- It must be aligned to ride straight, that is either perfectly symmetric, or with cancelling asymmetries. For example, the rear wheel can be oriented a little to one side, to reduce the tendency to turn to that side.
- It must have a freely turning front fork, the goal is frictionless, and without stiff shift or brake cables.

Finally, a down-slope may also be helpful to maintain speed.

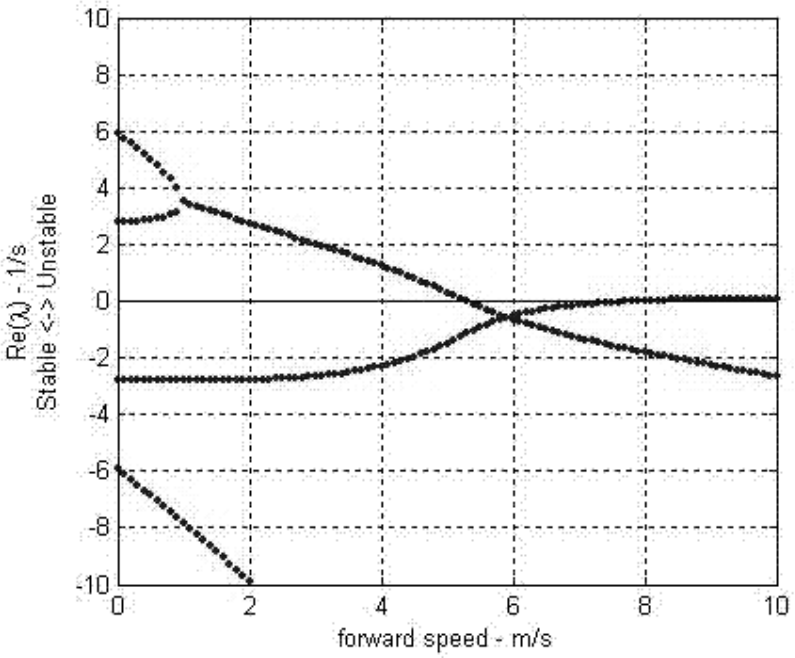
Follow these links to see the video: mpeg 3.7mb or mov 13.7mb

Please email us with questions or comments.

Copyright © 2003-2006 Schwab, Papadopoulos, Ruina, & Dressel, Delft University of Technology & Cornell University

JBike6 Eigenvalues

Eigenvalues



Eigenvalues are defined as the scalar quantities λ such that for matrix [A], if there is a non-zero vector x for which this equation [A]x = λx is true, then x is an **eigenvector** of [A] and λ is its corresponding eigenvalue. Another way to think of them is that if the matrix [A] is a transformation, then the eigenvectors of [A] are those vectors whose directions are not changed by that transformation, and the corresponding eigenvalues are how much the magnitudes or lengths of those eigenvectors are changed.

You can read more at Numerical Computing with MATLAB at The MathWorks and MathWorld at Wolfram Research.

Here is yet a third way to think of eigenvalues. First, consider an **eigenfunction**, the essence of which is 'solutions which evolve to remain self-similar'. Anything which evolves in such a fashion only changes magnitude, in proportion to current magnitude, i.e. exponential behavior. In a sense the determination of eigenfunctions is the key question, and the resulting eigenvalue can then be determined by plugging in the eigenfunction and noting the magnitude change [after one multiplication of a transition matrix, which could be based on an elapsed time interval].

In fact, you may consider the classical approach, given initially above, starting with eigenvalues, to be backwards. It is merely the trick that most quickly leads to finding the eigenfunctions, or indeed the eigenvalues after recognizing that the eigenvalues alone are needed to answer a stability question.

For the upright bicycle, any perturbation motion consists of time evolving lean and steer, with initial conditions on lean amount, lean rate, steer amount, and steer rate. The whole theory of eigenfunctions is that there is a complete set of special 'fundamental' motions (which evolve in self-similar fashion), and since the system is linear, any disturbance can be decomposed into fundamental motions, and its whole future evolution easily calculated when they are added together.

Any linear time-evolving, time-independent equations have the property that the sum of two solutions is also a solution. Therefore there is special value in looking for 'eigensolutions' - solutions that remain self-similar as they evolve, and hence evolve exponentially. Then any solution can be expressed as linear combinations of eigensolutions. For the purpose of studying stability, only the exponential growth, defined by the 'eigenvalues', is really needed.

So, eigenvalues are often found by relatively simple calculations, on the way to determining eigensolutions. For example when written in state space form, eigenvalues they are identified as 'expansion numbers' that make it possible to find self-similar (exponential) solutions, by the vanishing of a determinant (polynomial).

Another excellent on-line article about eigenvalues, eigenvectors, eigenspaces, and some of their applications can be found in Wikipedia.

Please email us with questions or comments.

Copyright © 2003-2006 Schwab, Papadopoulos, Ruina, & Dressel, Delft University of Technology & Cornell University

JBike6 Registration

JBike6 is free. But we would like to know who is using it and why. One issue, for example, is that years ago a disreputable lawyer claimed to use JBike5 (a version with errors) in a legal disposition. We will not use your name or email for any mailing lists. So feel free to enter your information, and then you can immediately download the program. Feedback of all kinds is not just welcome, but wanted.

Name:

Organization:

Email Address:

Reason for trying JBike6:

By checking this box you indicate that you have read and agree to the Terms and Conditions summarized below.

Yes

Submit Form

The license agreement goes basically like this:

- The Authors own all the intellectual property in the Software.
- The Authors grant to you a non-exclusive license to Use the Software.
- You may install and Use a copy of the Software on your compatible computer.
- Any copy of the Software that you make must contain the same copyright and other proprietary notices that appear on or in the Software.
- You may not, rent, lease, sublicense, assign or transfer your rights in the Software.

- The Software is being delivered to you "AS IS" and the Authors make no warranty as to its use or performance.
- THE AUTHORS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE.
- IN NO EVENT WILL THE AUTHORS BE LIABLE TO YOU FOR ANY DAMAGES, CLAIMS OR COSTS WHATSOEVER OR ANY CONSEQUENTIAL, INDIRECT, INCIDENTAL DAMAGES, OR ANY LOST PROFITS OR LOST SAVINGS
- If any part of the Agreement is found void and unenforceable, it will not affect the validity of the balance of the Agreement.
- JBike6 is a trademark of the Authors in the United States and/or other countries.

A full copy of the license agreement is included with the software and available on-line here.

Please email us with questions or comments.

Copyright © 2003-2006 Schwab, Papadopoulos, Ruina, & Dressel, Delft University of Technology & Cornell University

JBike6 End User License Agreement

NOTICE TO USER: PLEASE READ THIS CONTRACT CAREFULLY. BY USING ALL OR ANY PORTION OF THE JBIKE6 SOFTWARE ("SOFTWARE") YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT, INCLUDING, IN PARTICULAR THE LIMITATIONS ON: USE CONTAINED IN SECTION 2; TRANSFERABILITY IN SECTION 4; WARRANTY IN SECTION 6; AND LIABILITY IN SECTION 7. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU. THIS AGREEMENT IS ENFORCEABLE AGAINST YOU AND ANY LEGAL ENTITY THAT OBTAINED THE SOFTWARE AND ON WHOSE BEHALF IT IS USED. IF YOU DO NOT AGREE, DO NOT USE THIS SOFTWARE.

The Authors own all intellectual property in the Software. The Authors permit you to Use the Software only in accordance with the terms of this Agreement. Use of some third party materials included in the Software may be subject to other terms and conditions typically found in a separate license agreement or "Read Me" file located near such materials.

1. Definitions. "Software" means (a) all of the contents of the files, disk(s), CD-ROM(s) or other media with which this Agreement is provided, including but not limited to (i) computer information or software; (ii) related explanatory written materials or files ("Documentation"); and (b) upgrades, modified versions, updates, additions, and copies of the Software, if any, licensed to you by the Authors (collectively, "Updates"). "Use" or "Using" means to access, install, download, copy or otherwise benefit from using the functionality of the Software in accordance with the Documentation. "Computer" means an electronic device that accepts information in digital or similar form and manipulates it for a specific result based on a sequence of instructions. "The Authors" means Arend L. Schwab and Jim Papadopoulos.

2. Software License. As long as you comply with the terms of this Software License Agreement (this "Agreement"), the Authors grant to you a non-exclusive license to Use the Software for the purposes described in the Documentation.

2.1 General Use. You may install and Use a copy of the Software on your compatible computer.

2.2 Backup Copy. You may make one backup copy of the Software, provided your backup copy is not installed or used on any computer. You may not transfer the rights to a backup copy unless you transfer all rights in the Software as provided under Section 4.

2.3 Portable or Home Computer Use. In addition to the single copy permitted under Section 2.2, the primary user of the computer on which the Software is installed may make a second copy of the Software for his or her exclusive use on either a portable Computer or a Computer located at his or her home, provided the Software on the portable or home Computer is not used at the same time as the Software on the primary computer.

3. Intellectual Property Ownership, Copyright Protection. The Software and any authorized copies that you make are the intellectual property of and are owned by the Authors. The Software is protected by law, including without limitation the copyright laws of the United States and other countries, and by international treaty provisions. Except as expressly stated herein, this Agreement does not grant you any intellectual property rights in the Software and all rights not expressly granted are reserved by the Authors.

4. Restrictions.

4.1 Notices. You shall not copy the Software except as set forth in Section 2. Any copy of the Software that you make must contain the same copyright and other proprietary notices that appear on or in the Software.

4.2 Transfer. You may not, rent, lease, sublicense, assign or transfer your rights in the Software, or authorize all or any portion of the Software to be copied onto another user's computer except as may be expressly permitted herein. You may, however, transfer all your rights to Use the Software to another person or legal entity provided that: (a) you also transfer (i) this Agreement, and (ii) the Software and all other software or hardware bundled or pre-installed with the Software, including all copies, Updates and prior versions, to such person or entity; (b) you retain no copies, including backups and copies stored on a computer; and (c) the receiving party accepts the terms and conditions of this Agreement and any other terms and conditions upon which you legally purchased a license to the Software.

5. Updates. If the Software is an Update to a previous version of the Software, you must possess a valid license to such previous version in order to Use such Update. All Updates are provided to you on a license exchange basis. You agree that by Using an Update you voluntarily terminate your right to use any previous version of the Software. As an exception, you may continue to Use previous versions of the Software on your Computer after you Use the Update but only to assist you in the transition to the Update, provided that: (a) the Update and the previous versions are installed on the same computer; (b) the previous versions or copies thereof are not transferred to another party or Computer unless all copies of the Update are also transferred to such party or Computer; and (c) you acknowledge that any obligation the Authors may have

to support the previous versions of the Software may be ended upon availability of the Update.

6. NO WARRANTY. The Software is being delivered to you "AS IS" and the Authors make no warranty as to its use or performance. THE AUTHORS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE. EXCEPT FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT TO WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, THE AUTHORS MAKE NO WARRANTIES CONDITIONS, REPRESENTATIONS, OR TERMS (EXPRESS OR IMPLIED WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE) AS TO ANY MATTER INCLUDING WITHOUT LIMITATION NONINFRINGEMENT OF THIRD PARTY RIGHTS, MERCHANTABILITY, INTEGRATION, SATISFACTORY QUALITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. The provisions of Section 6 and Section 7 shall survive the termination of this Agreement, howsoever caused, but this shall not imply or create any continued right to Use the Software after termination of this Agreement.

7. LIMITATION OF LIABILITY. IN NO EVENT WILL THE AUTHORS BE LIABLE TO YOU FOR ANY DAMAGES, CLAIMS OR COSTS WHATSOEVER OR ANY CONSEQUENTIAL, INDIRECT, INCIDENTAL DAMAGES, OR ANY LOST PROFITS OR LOST SAVINGS, EVEN IF A REPRESENTATIVE OF THE AUTHORS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS, DAMAGES, CLAIMS OR COSTS OR FOR ANY CLAIM BY ANY THIRD PARTY. THE FOREGOING LIMITATIONS AND EXCLUSIONS APPLY TO THE EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION. THE AGGREGATE LIABILITY OF THE AUTHORS UNDER OR IN CONNECTION WITH THIS AGREEMENT SHALL BE LIMITED TO THE AMOUNT PAID TO THE AUTHORS FOR THE SOFTWARE, IF ANY.

8. Export Rules. You agree that the Software will not be shipped, transferred or exported into any country or used in any manner prohibited by the United States Export Administration Act or any other export laws, restrictions or regulations (collectively the "Export Laws"). In addition, if the Software is identified as export controlled items under the Export Laws, you represent and warrant that you are not a citizen, or otherwise located within, an embargoed nation (including without limitation Iran, Iraq, Syria, Sudan, Libya, Cuba, North Korea, and Serbia) and that you are not otherwise prohibited under the Export Laws from receiving the Software. All rights to Use the Software are granted on condition that such rights are forfeited if you fail to comply with the terms of this Agreement.

9. Governing Law. This Agreement will be governed by and construed in accordance with the substantive laws in force in the State of New York. The respective court of Tompkins County shall have non-exclusive jurisdiction over all disputes relating to this Agreement. This Agreement will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

10. General Provisions. If any part of this Agreement is found void and unenforceable, it will not affect the validity of the balance of this Agreement, which shall remain valid and enforceable according to its terms. This Agreement shall not prejudice the statutory rights of any party dealing as a consumer. This Agreement may only be modified by a writing signed by the Authors or their authorized representative. Updates may be licensed to you by the Authors with additional or different terms. This is the entire agreement between the Authors and you relating to the Software and it supersedes any prior representations, discussions, undertakings, communications or advertising relating to the Software.

11. Compliance with Licenses. If you are a business or organization, you agree that upon request from the Authors or their authorized representative, you will within thirty (30) days fully document and certify that use of any and all Software at the time of the request is in conformity with your valid licenses from the Authors.

If you have any questions regarding this Agreement or if you wish to request any information from the Authors please use the address and contact information included with this product to contact them.

JBike6 is a trademark of the Authors in the United States and/or other countries.

Please email us with questions or comments.

Copyright © 2003-2006 Schwab, Papadopoulos, Ruina, & Dressel, Delft University of Technology & Cornell University

JBike6 Download

Temporary format, until JBike6 is finalized.
Select a version to download below.

(22/09/06)
JBike6GUI.m & JBike6GUI.fig
-Correct some create_fcn names in JBike6GUI.fig
-Temporarily disable stability calculation

JBike6_2006_09_22.zip

(21/09/06)
JBike6GUI.m & JBike6GUI.fig
-Add label for check box to draw additional plots
-Impove name of menu items to increase or decrease size of symbols
-Calculate and warn user if bike will tip over forward
-Calculate 'max stability' and 'total stability' and display

JBrev.m
-Better control limits of plot in case of very large capsize speed
JBike6Batch.m
-Allow easier setting of parameters to vary
JBini_Benchmark_Paper.m
-New bike to calculate values in benchmark paper

JBike6_2006_09_21.zip

(13/09/06)
Subdirectory
-Remove GUI from name of JBike6 subdirectory
-include subdirectory in ZIP file

Readme.txt
-correct spelling of interface
-mention that MATLAB will run JBike6 from chosen subdirectory

JBike6.m
-Remove comments about running JBike6GUI.m

JBike6GUI.m & JBike6GUI.fig
-New button to toggle drawing additional plots
-Add menu options for 'bigger styles' and 'smaller styles'
-Change 'Styles' to 'Plot styles'
-Change 'Click for more information' to 'Help: click for more informaiton'
-Change 'Using JBike6...' to 'Help: using JBike6...'
-Add tooltips for 'U', 'V', and 'alpha'
-Fix copyright on separate 4 plot window'
-Use web command instead of dos command to launch help
-Replace blanks with zeros when user presses 'Calculate' button
-Better handle launching help on different system'

JBPlot4.m
-Update copyright statement

JBini.m
-Update version number

JBstar.m
-remove

JBhelp.htm
-Reflect changes above
-Move introduction before contents
-New Files list section
-New Architecture section
-Better explain about JBQRef.htm

JBQRef.htm
-Better explain about JBhelp.htm

JBike6Control.m
-New file to support plugging JBike6 into MATLAB's Control System Toolbox

JBike6_2006_09_13.zip

(12/03/06) Force directory of bike configurations to be absolute, not relative.
- use MATLAB's filesep function for transportability.
- also use absolute path for JBconfig.m
- do best to work with version 6.x as well

JBike6GUI_2006_03_12.zip

(06/02/06) Silence echo of waitbar handle
- Don't draw UV axis if not in UV mode
- Mention in help that alpha is always in XY
- Updated copyright noticed to 2006

JBike6GUI_2006_02_06.zip

(01/28/06) - Better support for MATLAB version 6.0.

JBike6GUI_2006_01_28.zip

(01/25/06) - Attempt to support MATLAB version 6.0.

JBike6GUI_2006_01_25.zip

(12/23/05) - Updated help.
- updated JBike6Batch.m

JBike6GUI_2005_12_23.zip

(12/20/05) - Alphabetize bikes when reading them in.
- save bike name, not number, in configuration file.
- clean file names: no \ / : * ? " < > | .
- fix bug with writing files with XY coords for front end.

JBike6GUI_2005_12_20.zip

(12/19/05) - more completely apply styles and settings to 4-plot window.

JBike6GUI_2005_12_19.zip

(12/18/05) - fix problem with CTRL-D after changing a parameter.

JBike6GUI_2005_12_18.zip

(12/17/05) - handle errors better, with a nice dialog.
- only calculate weave and capsize if user wants lines.

JBike6GUI_2005_12_17.zip

(12/16/05) - more parameter validity checks.
- new quick reference button.
- better tab order.
- more tool tip text.
- display wait bar during calculations.

JBike6GUI_2005_12_16.zip

(12/15/05) - inlude degrees or radians in unit text.
- Quick Reference Guide.

JBike6GUI_2005_12_15.zip

(12/12/05) - give standard gravity value in dialog.
- help updated with Jim''s comments.
- add tool tips for many controls.
- improve tab order between controls.

JBike6GUI_2005_12_12.zip

(12/11/05) - fix bug in finding 3rd double root.
- better split between real and imaginary colors in 4 plot window

JBike6GUI_2005_12_11.zip

(12/09/05) - Minor help updates: multiple complex pairs
- find 2nd pair of roots instead of warning about them in JBroots

JBike6GUI_2005_12_09.zip

(12/08/05) - Minor help updates: parameter limitations
- return from JBroots after warnings & handle 2nd root better

JBike6GUI_2005_12_08.zip

Items below are no longer available due to space limitations

(12/05/05) - Minor help updates
- fix a bug with ylabel of eigenvalue plot
- license agreement file: license.txt

JBike6GUI_2005_12_05.zip

(12/02/05) - More help updates

JBike6GUI_2005_12_02.zip

(12/01/05) - option to save eigenvalues to a file
- don't force min speed to be <= max speed, just ~=
- Help updated with Jim's extensive feedback

JBike6GUI_2005_12_01.zip

(11/15/05) - option to display legend on eigenvalue plot
- option to run in ''verbose'' mode or not
- updated accelerator keys
- updated help: finished glossary, added accelerator keys
- label colors or symbols better for real and imaginary

JBike6GUI_2005_11_15.zip

(11/11/05) - use user-setable symbols, line types, and sizes
- user selectable predefined black and white style
- option to use set styles in additional plot window
- reorient 3D plot to match the other three plots

JBike6GUI_2005_11_11pm.zip

11/10/05 - New Styles menu to set colors and dot sizes
- Save user setting in configuration file and load at startup
- fix bug in labeling wheelbase of backwards bike
- display more root digits
- keep name and version in single place: JBini.m
- remove double use of CTRL+p from menu system
- add interpolation method for finding double roots in eigenvalues
- display some status feedback in command window
- set plotting colors in JBini.m
- do not reset min, max and speed step when switching bikes
- check for invalid speeds better
- let user toggle between double root finding methods
- split JBdroot.m into JBdoubleroots.m and JBroots.m
- remove mention of colors from menu system
- rename module JBvroot.m to JBdroot.m
- added frequency of the weave motion as cyan dots
- add user control over directory bikes are read from and saved to

JBike6GUI_2005_11_10pm.zip

11/07/05 AED - only draw vertical axis if speed goes to 0 or below
- set xlim([min(speed) max(speed)])
- set ylim to handle negative speeds better
- add to on-line help
- uncommented JBini line in JBike6 and added warning to run JBike6GUI.m
- put call to JBike6GUI in JBike6 for force GUI anyway
- force guide to save GUI in "Ensure backward compatibility -v6"

JBike6GUI_2005_11_07.zip

11/06/05 AED - Let user toggle between UV and XY for front fork and basket
- Check for and complain about bad parameters: mass, inertia
- Tweak main window layout
- Change Swinn to Schwinn
- New menu shortcut keys CTRL+H for help, etc.
- New user parameter: front and rear wheel Ixx

- New user parameter: g
- Clean up what get's saved and restored from ini files
-- g, min_vel, max_vel, and vel_steps are stored but only as comments
-- even if read from ini file, they are overwritten with current GUI values
- New user parameter: minimum velocty
- Draw solid black vertical and horizontal axis
- Move on-line help to HTML
- Use green for weave speed: onset of stabilit'
- New ability to draw yellow lines at double roots
- New module: JBvroot.m
- New user parameter: velocity steps

JBike6GUI_2005_11_06.zip

(12/01/03) solved the problem of finding the stable speed interval in a unified and general way by introducing the so-called 'interval calculus'. This means that there are two new Matlab files:
-- JBpolyint.m: Which generates an interval list from a polynomial.
-- JBcombint.m: Which logically combines two interval lists.
- JBvcrit.m: Added the interval calculus. The result is the weave and/or capsize speed. Assign NaN if there is no weave and/or capsize speed.
- JBvcrit.m: Create an error when the determinant of the massmatrix is less than zero and create a warning when it equals zero.
- JBfig.m: Fixed the ''divide by zero'' warning in the case of zero mass of the front or rear wheel.
- JBfig.m: Draw a circle at the centre of mass representing 1/6 of the total mass.
- JBrev.m: Draw the lines representing the stable speed interval at the exact locations as given by vweave and/or vcapsize.
- JBike6GUI.m: Add a print facility to the file menu.

JBike6GUI_12_01_2003.zip

(10/09/03) Updated JBvcrit.m so that it returns NaN for vweave or vcapsize if they are not found.
- Includes JBike6Batch.m

JBike6GUI_10_09_2003.zip

(8/22/03) Updated JBvcrit.m so that it no longer reqires MATLAB's Symbolic Tool Box.
- Added Modification History menu option to Help menu

JBike6GUI_08_22_2003.zip

(8/18/03) Updated weave speed calculation with Arend's fix
- replaced both JBvcap.m and JBvweave.m with JBvcrit.m
- removed gravitational acceleration from GUI
- reconfigured controls on dialog to improve fit on small screens
- used dialog size and wheelbase to i

Please email us with questions or comments.

Copyright © 2003-2006 Schwab, Papadopoulos, Ruina, & Dressel, Delft University of Technology & Cornell University

Example of Benchmarking a Fully Non-linear Bicycle Model with JBike6

Introduction

The advent of cheap and plentiful computer power and numerical methods provides the opportunity to simulate bicycle motion from fully non-linear equations. Although systematic methods exist for generating these equations that avoid the complex assumptions and simplification required for linearized equations, the question of their accuracy remains.

Fortunately, a way exists to confirm fully non-linear equations: comparison with a known benchmark's eigenvalues. They are intrinsic to the modeled system and are completely independent of coordinates and units chosen.

JBike6 provides these eigenvalues from its linearized equations. Here we show how to extract them from the non-linear equations, and compare the results.

The Bicycle Model

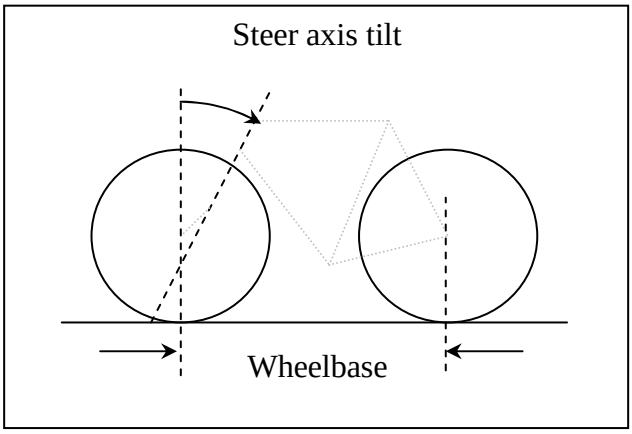
In both cases the bicycle model consists of:

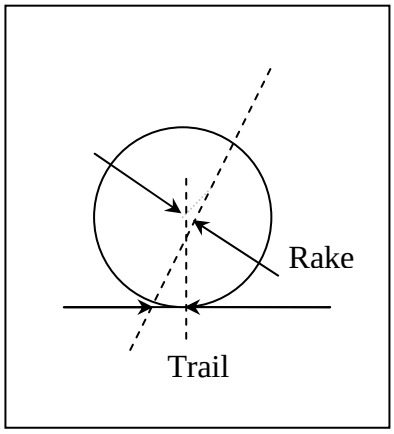
- 4 rigid bodies in 3 dimensions all with left-right mass symmetry.
- Connected by 3 revolute or pin joints: steering axis and two wheel hubs.
- Rider is rigidly attached to the frame (optional front and rear racks also rigidly attached).
- No friction at any joint.
- Wheels are knife-edged disks with left-right mass symmetry and polar mass symmetry, and they roll without slip or loss due to friction.
- Ground is a rigid, flat, horizontal surface.
- Only external forces are gravity, the ground reaction.
- Always starts in an upright, straight ahead configuration with some given forward velocity.
- Two non-holonomic constraints, one at each wheel contact point.

Key parameter definitions:

- **Wheelbase**: distance between wheel-ground contact points.

- **Steering axis tilt**: amount steering axis is tilted from the vertical.
- **Trial**: distance from front wheel ground contact point to point where steering axis intersects the ground.
- **Rake**: amount fork is offset, usually forward in order to reduce trail. (Redundant and not used, but included for completeness.)





Linearized Equations

Only three coordinates are necessary to specify a bicycle on a plane:

- Lean angle
- Steer angle
- Forward position of rear wheel.

However, the acceleration is taken to be zero to the first order and so the forward velocity is constant. This leaves only two non-trivial degrees of freedom: lean and steer angle. Two coupled second-order ordinary differential equations represent the motion of the system.[6[2]]

In the actual derivation for his Masters Thesis, Scott Hand used constrained Lagrange equations and then eliminated the constraints.

The Equations of Motion

$$M_{\theta\theta}\ddot{\theta}_r + K_{\theta\theta}\theta_r + M_{\theta\psi}\ddot{\psi} + C_{\theta\psi}\dot{\psi} + K_{\theta\psi}\psi = M_{\theta_r}$$

[6[2]] Hand Masters Thesis, Cornell, 1988, page 23.

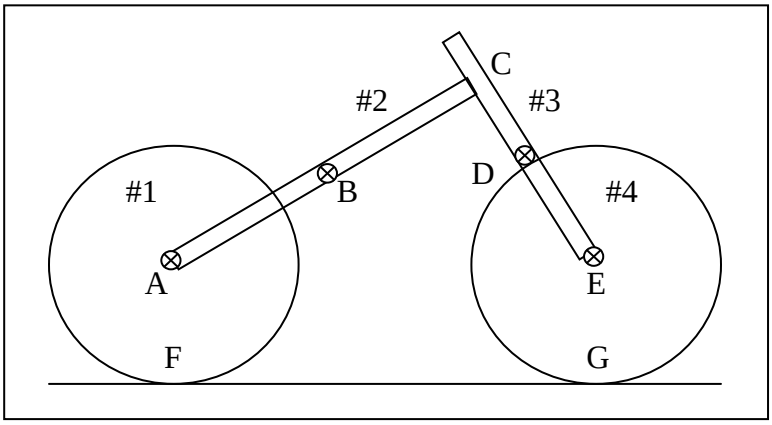
$$M_{\psi\psi}\ddot{\psi} + C_{\psi\psi}\dot{\psi} + K_{\psi\psi}\psi + M_{\psi\theta}\ddot{\theta}_r + C_{\psi\theta}\dot{\theta}_r + K_{\psi\theta}\theta_r = M_{\psi}$$

where[7[3]]

- θ_r is the lean angle of the rear assembly
- ψ is the steer angle of the front assembly relative to the rear assembly
- M_θ and M_ψ are moments (torques) applied at the rear assembly and the steering axis, respectively, and which are both taken to be zero.

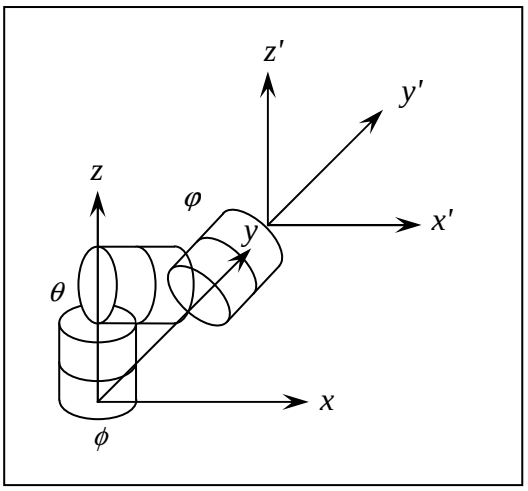
Non-linear Equations

The four rigid bodies used to model a bicycle (the same as for the linearized model) with the mass and inertia of the rear rack combined with that of the rear frame and the mass and inertia of the front basket combined with that of the front fork.



[7[3]] page 35.

The Euler Angles used to keep track of body orientation are φ, θ, and ψ:



The Coordinates are:

4 bodies in 3-D:

4 x 3 = 12 position coordinates

4 x 3 = 12 orientation coordinates (angles)

24 total coordinates

Constraints

3 revolute joints:

3 x 3 = 9 position coincident constraints

3 x 2 = 6 axis parallel constraints

2 ground contact points:

2 x 1 = 2 position constraints (vertical contact)

2 x 2 = 4 non-holonomic velocity constraints (wheels roll without slip)

17 coordinate constraints and 4 velocity constraints

The State Vector is:

$\left\lfloor \underset{\rightarrow}{r}_1, \underset{\rightarrow}{p}_1, \underset{\rightarrow}{r}_2, \underset{\rightarrow}{p}_2, \underset{\rightarrow}{r}_3, \underset{\rightarrow}{p}_3, \underset{\rightarrow}{r}_4, \underset{\rightarrow}{p}_4, \underset{\rightarrow}{v}_1, \underset{\rightarrow}{\omega'}_1, \underset{\rightarrow}{v}_2, \underset{\rightarrow}{\omega'}_2, \underset{\rightarrow}{v}_3, \underset{\rightarrow}{\omega'}_3, \underset{\rightarrow}{v}_4, \underset{\rightarrow}{\omega'}_4 \right\rfloor$ where

$\underrightarrow{r}_i$ is the 3-D position vector of body *i* in space-fixed coordinates: (x_i, y_i, z_i)

$\underrightarrow{p}_i$ is the orientation vector of body *i* in space-fixed Euler angles: $(\phi_i, \theta_i, \varphi_i)$

$\underrightarrow{v}_i$ is the 3-D velocity vector of body *i* in space-fixed coordinates

$\underrightarrow{\omega'}_i$ is the angular velocity vector of body *i* in body-fixed coordinates

The Constrained Equations of Motion are:

$$\begin{bmatrix} \begin{bmatrix} m_i\mathbf{I}_3 & \\ & \mathbf{J}'_i \end{bmatrix}_{24\times24} & \left[\mathbf{D}^T\right]_{24\times21} \\ \left[\mathbf{D}\right]_{21\times24} & \left[\mathbf{0}\right]_{21\times21} \end{bmatrix} \begin{bmatrix} \dot{\underrightarrow{v}}_i \\ \dot{\underrightarrow{\omega}}_i \\ \lambda_1 \\ \vdots \\ \lambda_{21} \end{bmatrix} = \begin{bmatrix} \sum \underrightarrow{f}_i \\ \sum \bar{M}'_i - \underrightarrow{\omega'}_i \times \left(\mathbf{J}'_i \underrightarrow{\omega'}_i\right) \\ \\ g_i \end{bmatrix}$$

where **D** is the 'Jacobian' of the velocity constraints and g_i are the 'convective' terms, and since $\mathbf{I}_3$ is used to represent the 3x3 identity, the body-fixed inertia matrix for body *i* is written $\mathbf{J}'_i$.[8[4]]

Numerical Implementation

Written in MATLAB version 6.5 from The MathWorks

With numerical integration via RK4, although integration is not used in the calculation of eigenvalues.

And adjusting positions and velocities back to 'constraint surface' after each numerical integration step:

- Adjust positions by iterating with Coordinate Projection Method
- Minimize error and smooth results by forcing at least one iteration. Without this technique, error in the position constraints builds over time and then requires a 'jarring' correction that shows up in the running energy calculations. It appears that this technique better keeps the error in position constraints under control so that multiple steps are rarely, if ever, required.
- Adjust velocities with Pseudo inverse in one step.

[8[4]] Schwab class notes, 2003

Parameters

Use in both implementations: 'Schwinn Crown'. Units are meters, kilograms, and radians.

```
wheelbase = 1.01600000;
head_angle = 1.23879599;
trail = 0.09096800;

% rear wheel
Drearwheel = 0.68580000;
mrearwheel = 1.81818200;
Irearwheel = [0.08551300 0.08551300 0.17102600];
alphaIrearwheel = 0.00000000;

% front wheel
Dfrontwheel = 0.68580000;
mfrontwheel = 1.81818200;
Ifrontwheel = [0.08551300 0.08551300 0.17102600];
alphaIfrontwheel = 0.00000000;

% rider
mrider = 80.00000000;
xcmrider = 0.30000000;
ycmrider = 1.20000000;
Irider = [10.53112900 2.46887100 12.00000000];
alphaIrider = -0.25957299;

% frame
mframe = 2.50000000;
xcmframe = 0.30000000;
ycmframe = 0.50000000;
Iframe = [0.05857900 0.34142100 0.40000000];
alphaIframe = 0.39269899;

% front fork
mfork = 1.50000000;
ucmfork = 0.00000000;
vcmfork = 0.68580000;
Ifork = [0.05879000 0.00058800 0.05879000];
alphaIfork = 0.33200000;

% front basket
mbasket = 1.50000000;
ucmbasket = 0.15000000;
vcmbasket = 0.80000000;
Ibasket = [0.01000000 0.01000000 0.01000000];
alphaIbasket = 0.00000000;

% rear rack
mrack = 2.00000000;
xcmrack = -0.10000000;
ycmrack = 0.80000000;
Irack = [0.02000000 0.02000000 0.02000000];
```

```
alphaIrack = 0.00000000;
```

Eigenvalues from linearized equations

Rewrite the equations of motion in matrix form:

$$\left[\overbrace{\begin{pmatrix} M_{\theta\theta} & M_{\theta\psi} \\ M_{\psi\theta} & M_{\psi\psi} \end{pmatrix}}^{M} D^2 + \overbrace{\begin{pmatrix} C_{\theta\theta} & C_{\theta\psi} \\ C_{\psi\theta} & C_{\psi\psi} \end{pmatrix}}^{C} D + \overbrace{\begin{pmatrix} K_{\theta\theta} & K_{\theta\psi} \\ K_{\psi\theta} & K_{\psi\psi} \end{pmatrix}}^{K}\right] \begin{pmatrix} \theta_r \\ \psi \end{pmatrix} = \begin{pmatrix} M_{\theta_r} \\ M_{\psi} \end{pmatrix}$$

where *M* is the mass matrix, *C* is the damping matrix, *K* is the stiffness matrix, and *D* is the differential operator. Note that for the upright bike $C_{\theta\theta} = 0$, and so it did not appear above in the equations of motion.

Combine the *M*, *C*, and *K* matrices into 2 4x4 matrices, *A* and *B*,

$$B = \begin{bmatrix} M_{\theta\theta} & M_{\theta\psi} & 0 & 0 \\ M_{\psi\theta} & M_{\psi\psi} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } A = \begin{bmatrix} -C_{\theta\theta} & -C_{\theta\psi} & -K_{\theta\theta} & -K_{\theta\psi} \\ -C_{\psi\theta} & -C_{\psi\psi} & -K_{\psi\theta} & -K_{\psi\psi} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and calculate the 4 *generalized eigenvalues* of them via:

$Ax = \lambda Bx$

Where the values of λ that satisfy the equation are the generalized eigenvalues and the corresponding values of *x* are the generalized right eigenvectors.

In MATLAB:

```
B = [M zeros(2); zeros(2) eye(2)];
A = [-C -K; eye(2) zeros(2)];
[V,D] = eig(A, B);
```
[9[5]]

Eigenvalues from non-linear equations

Represent the system in state-space form as

$$\dot{\vec{x}} = [\mathbf{A}]\vec{x}$$

and assume a solution of the form $x = C_1 e^{[\mathbf{A}]t}$.

[9[5]] Papadopoulos and Schwab JBike6, 2003

Then the eigenvalues, λ, of the matrix [**A**] describe stability: stable for $\lambda < 0$, unstable otherwise.

To construct this matrix [**A**] for a nonlinear system being solved numerically, recognize that the function to compute the accelerations (called by the ode solver, RK4 in this case) has the form $\dot{\vec{x}} = f(\vec{x})$.[10[6]]

Thus the matrix we seek is merely $[\mathbf{A}] = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n} \right]$.

However, $f(\vec{x})$ is not some function that we can simply differentiate symbolically.

Instead, to calculate partial derivatives numerically, use the definition of the derivative:

$$f'(x) = \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$ (for forward differencing, with error O(ε))

$$f'(x) = \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon}$$ (or even better: centered differencing, with error O(ε^2))

If access to calculated accelerations is available (which in this case is the output of the function called by the ode solver) then

$$[\mathbf{A}] = \left[\frac{f(x_0 + \varepsilon\hat{e}_1) - f(x_0)}{\varepsilon} \quad \frac{f(x_0 + \varepsilon\hat{e}_2) - f(x_0)}{\varepsilon} \quad \cdots \quad \frac{f(x_0 + \varepsilon\hat{e}_n) - f(x_0)}{\varepsilon} \right]$$

In order to select the best value for epsilon, we simply iterate until convergence.

The value of epsilon for each state variable and forward speed is chosen in a loop in which it is decimated until the resulting candidate values for a column in the matrix converge. The best value turns out to be 1e-11.

The eigenvalues of this matrix [**A**] can be compared to the eigenvalues from the linearized equations. In this particular case, because the non-linear equations use 48 state variables instead of the 4 used in the linearized equations to represent the 2 non-trivial degrees of freedom in the system, 48 values are generated. 4 of them correspond to the 2 non-trivial degrees of

---

[10[6]] Ruina, 2004

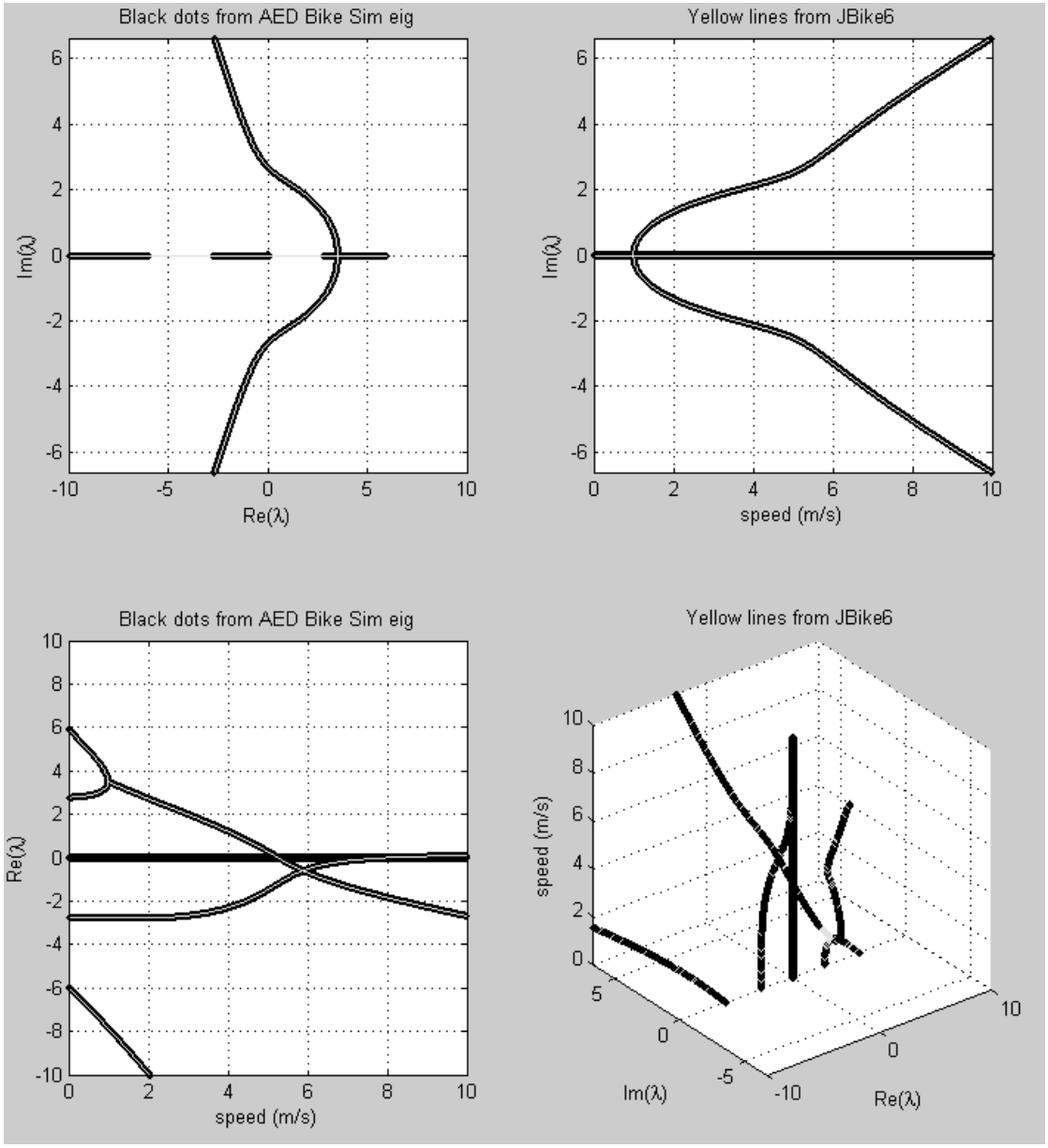
freedom in the system, and can be compared to the 4 generated from the linearized equations.

In MATLAB:

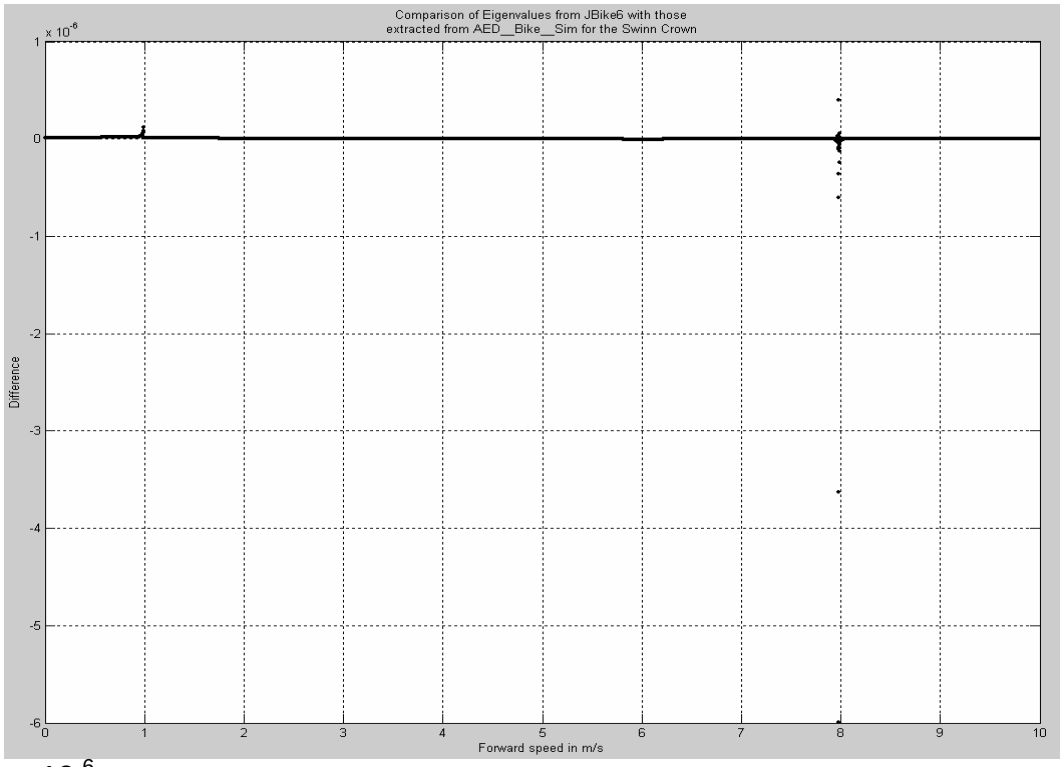
```
v0 = v0_min;                % start with the first forward speed
speed = 1;                  % index this as number 1
while v0 <= v0_max          % while there are more forward speeds
to use
  for sv = 1: state_variables  % for each state variable
    e = zeros(state_variables, 1);  % initialize an epsilon vector
    e(sv, 1) = epsilon;             % set one value to non-zero
    for direction = -1:2:1          % for each dir for center diff
      z(:, n) = z_0 + (epsilon .* direction);  % add epsilon to st
v
      % Apply constraints to the four bodies
      %   iterate to adjust current positions subject to
constraints
      %   adjust current velocities subject to constraints
      % Calculate accelerations (Left Hand Side)/(Right Hand Side)
      [z(:, n+1), lambdas(:, n+1)] = ode(z(:, n), M, Mo_b, f,
r_b);
      if direction == -1
        f_z_0_m_e = z(:, n+1);  % f(z_0 – epsilon)
      else
        f_z_0_p_e = z(:, n+1);  % f(z_0 + epsilon)
        A(:, sv) = (f_z_0_p_e - f_z_0_m_e) / (2 * epsilon);  % add
      end
    end
  end
  [V,D] = eig(A);           % calculate eigenvalues
  eigs(:, speed) = diag(D); % store them away
  v0s(speed) = v0;          % store forward speed
  v0 = v0 + v0_step;        % increment forward speed
  speed = speed + 1;        % increment index
end
```

Results

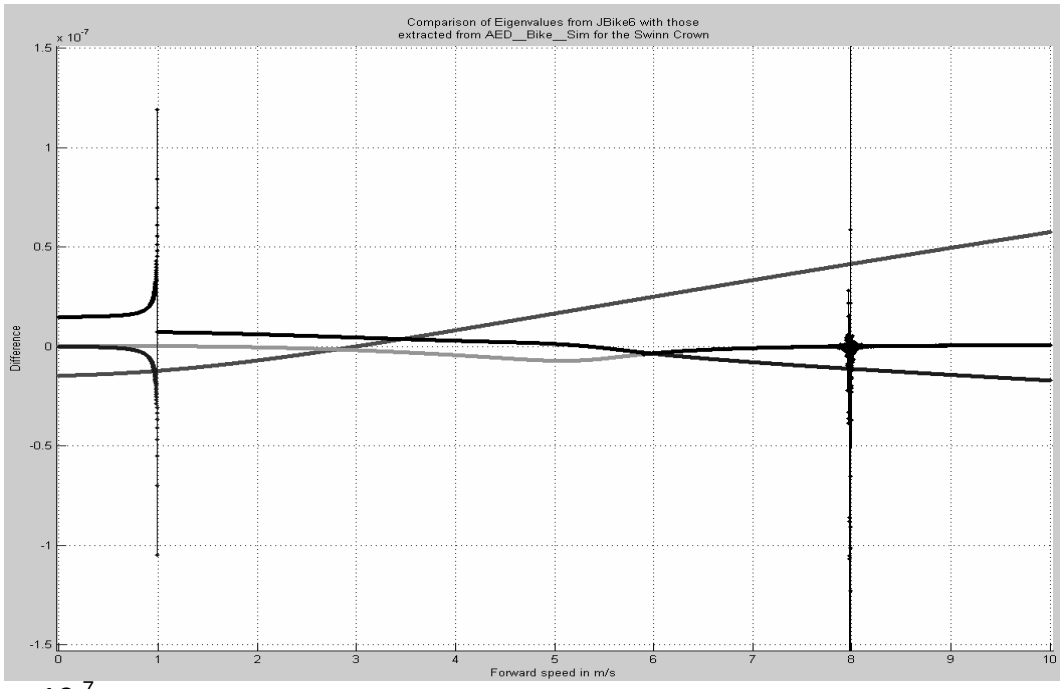
The plots below show the excellent match of both real and imaginary parts of eigenvalues from the two methods.



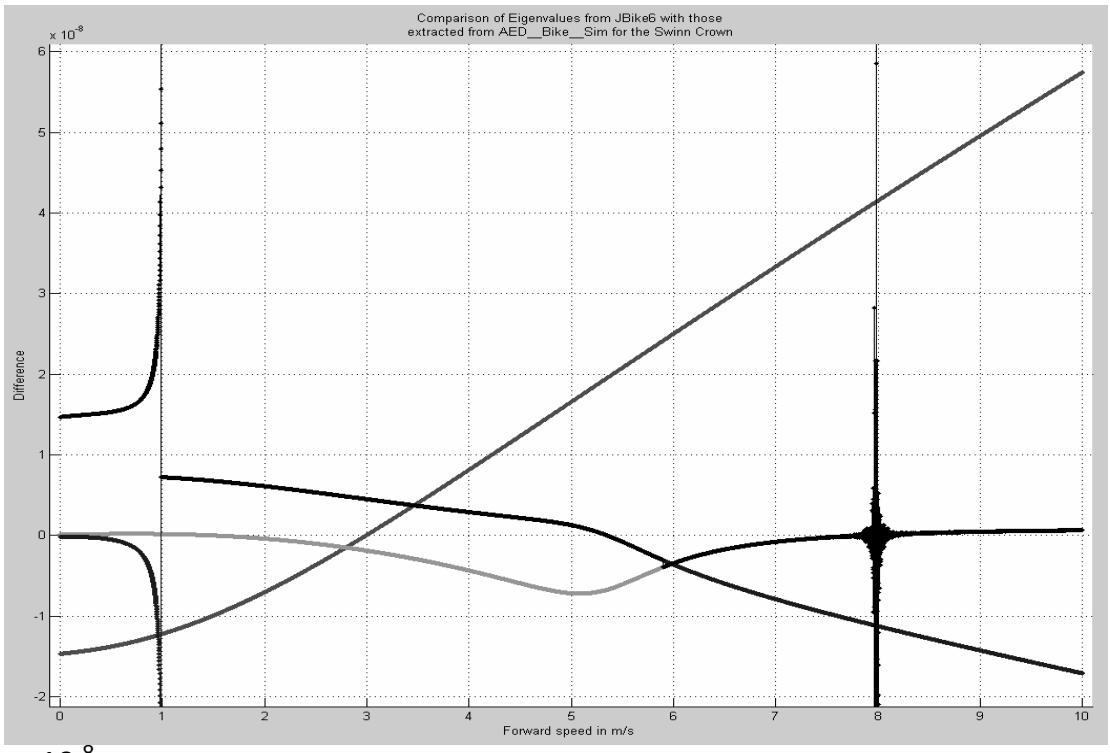
The plots below show the difference between just the real parts of the values from the two methods.



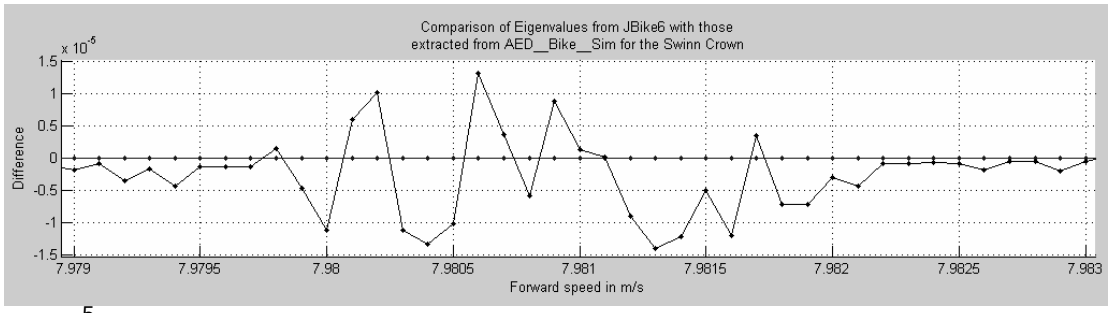
at 10^{-6}



at 10^{-7}



at 10^{-8}



at 10^{-5} and only the part of the forward speed range near the capsize speed.

Maximum difference between the two methods occurs near the capsize speed (7.9807 m/s), where an eigenvalue crosses the real axis from negative to positive. The difference peaks at just under 1.5e-05.

Another, smaller increase in error occurs just before (forward speeds < 1 m/s) a double root: a bifurcation where two eigenvalues with different real parts and zero imaginary parts approach the same value and then (for higher forward speeds) have the same real parts and complimentary imaginary parts. There the difference peaks at just over 1e-07.

In all other cases, the difference between eigenvalues from the two different methods remains less than 6e-08.

Conclusions

The non-linear equations of motion are confirmed.

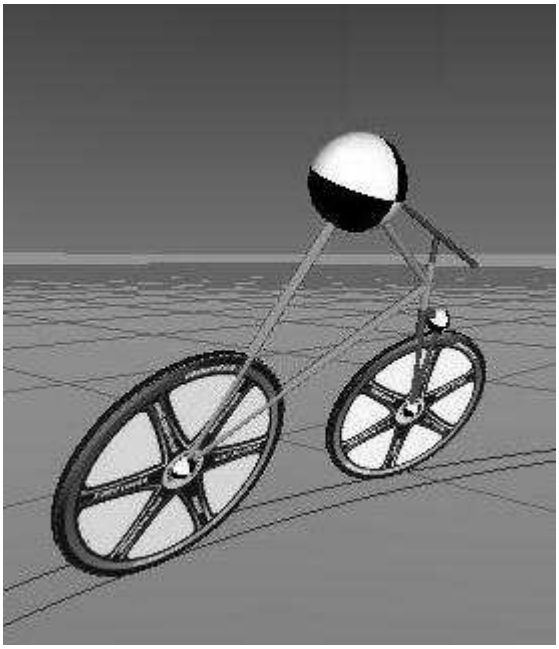
Future Work

Investigate and eliminate spike in difference between eigenvalues from linearized and non-linear equations when velocity is near 8 m/s.

Investigate generating more accurate eigenvalues from non-linear model with more accurate derivatives.

Simulation

Here are some video clips of the animated simulation confirmed by JBike6.



Example of Applying Control Theory to Bicycle Linearized Equations of Motion as Implemented in JBike6

Introduction

Most control theory techniques require linearized equations of motion of the system to be controlled. JBike6 already contains these, and all that is needed is to organized them into the required form.

Linearized Equations of Motion

Here are the linearized equations of motion for an uncontrolled bicycle as implemented in JBike6:

```
M0*qdd+(C1.*v)*qd+(K0+K2.*v^2)*q=0
    q=[lean angle, steer angle]
```

$$\text{i.e } \mathbf{M0}\ddot{q} + \mathbf{C1}v\dot{q} + \left[\mathbf{K0} + \mathbf{K2}v^2\right]q = 0$$

where

$$q = \begin{bmatrix} \text{lean angle} \\ \text{steer angle} \end{bmatrix}$$

v = forward speed

M0 = Mass matrix

C1 = Velocity sensitivity matrix

K0 and **K2** = Stiffness matrices

Introduce the *u(t)* and *y(t)* of control theory

Let the control 'input' $u(t)$ be steer torque τ and the 'output' $y(t)$ be lean angle. Then

$$\mathbf{M0}\ddot{q} + \mathbf{C1}v\dot{q} + \left[\mathbf{K0} + \mathbf{K2}v^2\right]q = u(t)$$

Multiply through by $\mathbf{M0}^{-1}$ now, to avoid inverting a singular matrix later

$$\ddot{q} + \mathbf{M0}^{-1}\mathbf{C1}v\dot{q} + \mathbf{M0}^{-1}\left[\mathbf{K0} + \mathbf{K2}v^2\right]q = \mathbf{M0}^{-1}u(t)$$

For computational simplicity, let

$$\mathbf{M} = \mathbf{M0}^{-1}, \quad \mathbf{S} = \mathbf{M0}^{-1}\mathbf{C1}v, \quad \text{and} \quad \mathbf{K} = \mathbf{M0}^{-1}\left[\mathbf{K0} + \mathbf{K2}v^2\right]$$

$$\ddot{q} + \mathbf{S}\dot{q} + \mathbf{K}q = \mathbf{M}u(t)$$

Represent in state-space form

To represent in state-space form, $\dot{\vec{x}} = \mathbf{A}\vec{x} + \mathbf{B}u$, and $y = \mathbf{C}\vec{x} + \mathbf{D}u$, recast from 2 coupled 2nd-order equations to 4 1st-order equations. To that end, let $x_1(t) = q_1(t)$, $x_2(t) = \dot{q}_1(t)$, $x_3(t) = q_2(t)$, and $x_4(t) = \dot{q}_2(t)$. Thus

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \mathbf{S}\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \mathbf{K}\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \mathbf{M}\begin{bmatrix} 0 \\ \tau \end{bmatrix}$$

becomes

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 & 0 \\ \mathbf{K}_{11} & \mathbf{S}_{11} & \mathbf{K}_{12} & \mathbf{S}_{12} \\ 0 & 0 & 0 & -1 \\ \mathbf{K}_{21} & \mathbf{S}_{21} & \mathbf{K}_{22} & \mathbf{S}_{22} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \mathbf{M}_{11} & 0 & \mathbf{M}_{12} \\ 0 & 0 & 0 & 0 \\ 0 & \mathbf{M}_{21} & 0 & \mathbf{M}_{22} \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ \tau \end{bmatrix},$$

which can be written as

$$\dot{\vec{x}} + \tilde{\mathbf{K}}\vec{x} = \tilde{\mathbf{M}}\vec{u} \text{ or } \dot{\vec{x}} = -\tilde{\mathbf{K}}\vec{x} + \tilde{\mathbf{M}}\vec{u}$$

So, finally

$$\mathbf{A} = -\tilde{\mathbf{K}} = -\begin{bmatrix} 0 & -1 & 0 & 0 \\ \mathbf{K}_{11} & \mathbf{S}_{11} & \mathbf{K}_{12} & \mathbf{S}_{12} \\ 0 & 0 & 0 & -1 \\ \mathbf{K}_{21} & \mathbf{S}_{21} & \mathbf{K}_{22} & \mathbf{S}_{22} \end{bmatrix},$$

$$\mathbf{B} = \tilde{\mathbf{M}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \mathbf{M}_{11} & 0 & \mathbf{M}_{12} \\ 0 & 0 & 0 & 0 \\ 0 & \mathbf{M}_{21} & 0 & \mathbf{M}_{22} \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \text{ and}$$

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

Convert to a transfer function in MATLAB

Use MATLAB's `ss2tf()` function to convert the above to the numerator and denominator of a transfer function:

```
[numG, denG] = ss2tf(A,B,C,D)
```

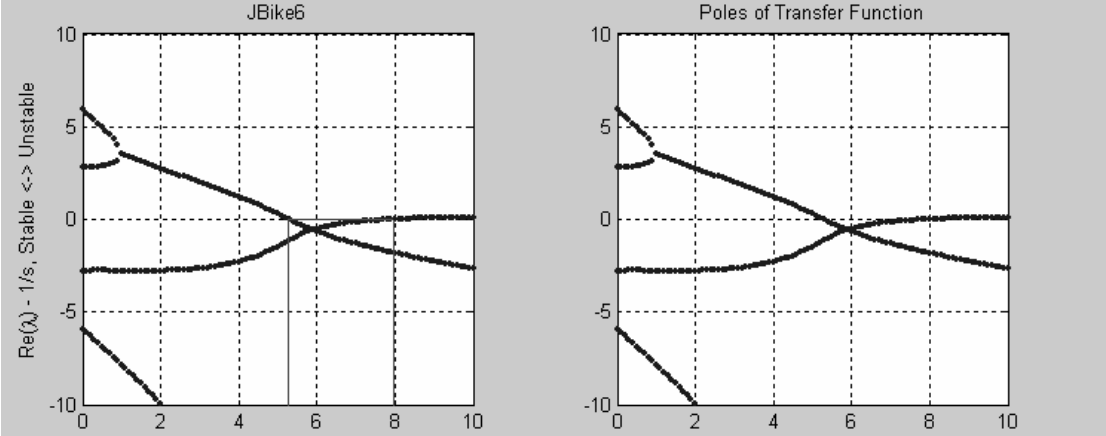
Then we can combine them into a single transfer function with MATLAB's `tf()` function:

```
G = tf(numG, denG)
```

Verify Results

Compare the Pole-zero map of the resulting transfer function to the eigenvalues already produced by JBike6

For the Swinn Crown, the plots look like this:



The MATLAB source code:

```
% From JBike6
% The linearized equations of motion read:
%
%       M0*qdd+(C1.*v)*qd+(K0+K2.*v^2)*q=0
%
% where v is the forward speed of the bike.

% set the bicycle parameters
JBini

% calculate the linearized equations of motion
JBmck

% calculate the weave and capsize speed
JBvcrit

% calculate the eigenvalues and eigenvectors for a speed range
JBeig

% plot only the Real part of the eigenvalues
subplot(1,2,1);
JBrev
title('JBike6'); a = axis;
drawnow;

% create transfer function from state-space representation
% use x1 = q1, x2 = q1_dot, x3 = q2, x4 = q2_dot
subplot(1,2,2);
v = 0:0.1:10;
M = inv(M0);
for i = 1: length(v)
    S = M*(C1.*v(i));
    K = M*(K0 + K2.*v(i)^2);
    A = -[0,      -1,     0,      0;
          K(1,1), S(1,1), K(1,2), S(1,2);
          0,      0,      0,      -1;
          K(2,1), S(2,1), K(2,2), S(2,2)];

    B = [0, 0,      0, 0;
         0, M(1,1), 0, M(1,2);
         0, 0,      0, 0;
         0, M(2,1), 0, M(2,2)];

    C = [1, 0, 0, 0];
    D = [0, 0, 0, 0];
    [numG, denG] = ss2tf(A, B, C, D, 2);
    G = tf(numG, denG);
    [p, z] = pzmap(G);
    plot(v(i).*ones(1, length(p)), p, '.'); hold on;
end
grid on; hold off; axis(a); title('Poles of Transfer
Function');
```

REFERENCES

Fajans, Joel. Email Questions and Answers
socrates.berkeley.edu/~fajans/Teaching/MoreBikeFiles/BikeQuestions.htm

Fajans, Joel (July 2000). "Steering in bicycles and motorcycles". American Journal of Physics 68 (7): 654–659.
socrates.berkeley.edu/~fajans/pub/pdffiles/SteerBikeAJP.PDF

Hand, Richard S. (1988). Comparisons and Stability Analysis of Linearized Equations of Motion for a Basic Bicycle Model (PDF).
ruina.tam.cornell.edu/research/topics/bicycle_mechanics/papers/comparisons_stability_analysis.pdf

Jones, David E. H. (1970). "The stability of the bicycle". Physics Today 23 (4): 34–40.

Klein, Richard E.; et al. Bicycle Science.
www.losethetrainingwheels.org/default.aspx?Lev=1&ID=20

Meijaard, J.P., Papadopoulos, J.M., Ruina, A. , and Schwab, A.L. (2006) Linearized dynamics equations for the balance of a bicycle: a benchmark,
ruina.tam.cornell.edu/research/topics/bicycle_mechanics/papers/BicyclePaper1Andyv31.pdf

Moler, Cleve (2004) Numerical Computing with MATLAB
www.mathworks.com/moler/chapters.html

Neĭmark, J. I. and Fufaev, N. A., Dynamics of Nonholonomic Systems, (American Mathematical Society Translations of Mathematical Monographs, Vol. 33, 1972, pp. 330-374.

Ruina, Andy; Rudra Pratap (2002). Introduction to Statics and Dynamics (PDF), Oxford University Press, 350.

Schwab, Arend L., Jaap P. Meijaard, Jim M. Papadopoulos (2005). "Benchmark Results on the Linearized Equations of Motion of an Uncontrolled Bicycle" (PDF). KSME International Journal of Mechanical Science and Technology 19 (1): 292–304.

The MathWorks MATLAB help
www.mathworks.com/support/product/product.html?product=ML

Whitt, Frank R.; David G. Wilson (1982). Bicycling Science, Second edition, Massachusetts Institute of Technology, 198–233. ISBN 0-262-23111-5.

Wolfram Research MathWorld mathworld.wolfram.com