Maple and the Parachute Problem: Modelling with an Impact

*Douglas B. Meade**

Abstract

The "parachute problem" involves a first-order ODE in which the coefficient of air resistance is a piecewise-defined function. This paper focuses on the use of Maple's symbolic, numeric, and graphic tools to facilitate the analysis of this initial value problem. Of particular interest is a qualitative analysis of the solution. The form of the problem most commonly found in textbooks is shown to be physically unrealistic. An improved model is proposed and analyzed. The paper concludes with the analysis of parameter sensitivity of the model and the solution of an interesting control problem. The graphical analysis is based on two Maple procedures, `motion` and `plot_motion`, which are defined in this paper.

Introduction

The "parachute problem" is based on a model for the motion of a skydiver when the coefficient of air resistance changes between free-fall and final descent. The author has previously shown that many of the problems posed in textbooks are not physically realistic and has proposed an improved — but not more complicated — based on real-life information about skydiving[6]. While the basic problems are not too sophisticated, manual computation of the solution is prohibitive. As a result, much of the analysis will be based on information obtained directly from the differential equation. The purpose of this paper is to illustrate some of the ways in which Maple can be used to facilitate the analysis; see [6] for a more mathematical perspective of the problem.

The Original Model

The "parachute problem" will be considered as a system of two initial value problems for the position, x, and (vertical) velocity, v, of a skydiver under the forces of gravity and air resistance. When the force due to air resistance is proportional to the velocity of the parachutist the model is:

$$\begin{array}{rcll} v' & = & -g - \frac{k}{m}v, & v(0) = 0, \\ x' & = & v, & x(0) = x_0 \end{array} \tag{1}$$

Reference	k_1/m	k_2/m	t_d	x_0
[1, p. 141, #20]	1/6	5/3	60sec	n/a
[3]	4/15	4/3	$x(14) = 0$	1200ft
[4, p. 95, #10]	3/20	3/2	20sec	10,000ft
[5, p. 109, #20]	1/5	≈ 1.56	$v(t_d) = v^*$	2000m
[9, p. 112, Ex. 3]	1/5	7/5	60sec	4000m

Table 1: Coefficients of air resistance, deployment criteria, and jump height found in different versions of PP.

where g is the gravitational constant, k is the coefficient of air resistance, m is the mass of the skydiver, and x_0 is the altitude at which the jump begins. The coefficient of air resistance is frequently modelled as a piecewise constant function ([1], [3], [4], [5], [9]):

$$k(t) = \begin{cases} k_1, & t < t_d \\ k_2, & t \geq t_d \end{cases} \tag{2}$$

where t_d is the time when the parachute is deployed.[1] Parameter values found in a variety of textbooks are summarized in Table 1.

To prepare for the Maple analysis, the `plots` package is needed:

```
> restart; with( plots ):
```

The basic problem to be studied is

```
> SYS := { diff( v(t), t ) = -g - k/m*v(t),
>          diff( x(t), t ) = v(t) };
```

$$SYS := \left\{ \frac{\partial}{\partial t}\,\mathrm{v}(\,t\,) = -g - \frac{k\,\mathrm{v}(\,t\,)}{m}, \frac{\partial}{\partial t}\,\mathrm{x}(\,t\,) = \mathrm{v}(\,t\,) \right\}$$

with initial conditions

```
> IC := { v(0) = 0, x(0) = x0 };
```

$$IC := \{\,\mathrm{x}(\,0\,) = x0, \mathrm{v}(\,0\,) = 0\,\}$$

As the system of ODEs and initial conditions are both sets, it will be useful to have a means to extract a specific entry from either object.

```
> ic := (IVP,y0) -> op( select( has, IVP, y0 ) ):
> ode := (IVP,y) ->
>        op( select( has, IVP, diff(y(t),t) ) ):
```

The position (height above ground level), velocity, acceleration, and jerk (rate of change of acceleration)

*Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA; E-mail: `meade@math.sc.edu`; WWW URL: `http://www.math.sc.edu/~meade/`

[1] Note that t_d may be specified implicitly, *e.g.*, deployment occurs when the velocity is 95% of the terminal free-fall velocity [5], in which case the IVP may not be linear.

provide a good description of the motion of the skydiver. Given an initial value problem in terms of x and v, the procedure `motion` uses `dsolve,numeric` to compute the motion of the object at user-specified times; `plot_motion` uses the information computed by `motion` to creates a single plot containing any combination of the position, velocity, acceleration, and jerk.

```
> motion := proc( IVP:set(equation),
>                TIMES:list(numeric), t )
>   local POS, VEL, ACC, JERK, pts, acc, jerk,
>         soln, T;
>   options `Copyright 1996 by Douglas B. Meade`;
>   soln:=dsolve(IVP, {x(t),v(t)}, type=numeric,
>                output=listprocedure):
>   POS:=subs( soln, x(t) );
>   VEL:=subs( soln, v(t) );
>   acc:=solve( ode(IVP,v), diff(v(t),t) );
>   ACC:=unapply( subs( v(t)='VEL(t)', acc ), t );
>   jerk:=diff(acc,t);
>   JERK:=unapply(subs( {v(t)='VEL(t)',
>                        'diff(v(t),t)'='ACC(t)'},
>                        jerk ), t );
>   pts:=seq(
>         evalf([T,POS(T),VEL(T),ACC(T),JERK(T)]),
>         T=sort(TIMES) ):
>   RETURN( [ pts ] );
> end:
> plot_motion :=
>   proc( pts:list, scale:set(equation) )
>   local P, PLOTS, S, VARS;
>   options `Copyright 1996 by Douglas B. Meade`;
>   PLOTS := NULL;
>   VARS := map(lhs,scale);
>   if member(X,VARS) then
>     S:=subs(scale,X);
>     PLOTS:=PLOTS,
>            plot([ seq( [P[1],P[2]/S], P=pts )],
>            color = RED ):
>   fi;
>   if member(V,VARS) then
>     S:=subs(scale,V);
>     PLOTS:=PLOTS,
>            plot([ seq( [P[1],P[3]/S], P=pts )],
>            color = BLUE ):
>   fi;
>   if member(A,VARS) then
>     S:=subs(scale,A);
>     PLOTS:=PLOTS,
>            plot([ seq( [P[1],P[4]/S], P=pts )],
>            color = GREEN ):
>   fi;
>   if member(J,VARS) then
>     S:=subs(scale,J);
>     PLOTS:=PLOTS,
>            plot([ seq( [P[1],P[5]/S], P=pts )],
>            color = CYAN ):
>   fi;
>   RETURN( display( { PLOTS } ) );
> end:
```

Using Maple's `Heaviside` function to implement the piecewise-constant air resistance (2),

```
> K[pw_const] :=  k1+(k2-k1)*Heaviside(t-td):
```

the motion of the skydiver is governed by the initial value problem

```
> IVP := subs( k=K[pw_const], SYS ) union IC:
```

The specific motion can be produced after specifying values for each parameter. The values given in [9] are

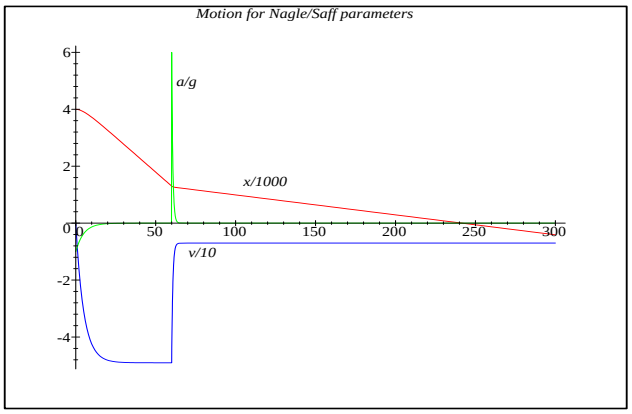
```
> PARAM[ns] := k1=15, k2=105, td=60, x0=4000,
>              m=75, g=981/100:
```

Lastly, the list of times is selected with a finer resolution near the time of deployment:

```
> snapshot[ns] := [ $(0..58),
>                   i/10 $ i = 590 .. 649,
>                   5*i $ i = 13 .. 60 ]:
```

An informative graph of the motion during the first five minutes of the jump is obtained as follows:

```
> PTS[ns]:=motion( subs(PARAM[ns],IVP),
>                  snapshot[ns], t ):
> scale[noJ] := { 'X'=1000, 'V'=10, 'A'=981/100 }:
> display(
>   {
>     plot_motion( PTS[ns], scale[noJ] ),
>     textplot( [ [ 120, 1.5, `x/1000` ],
>                 [ 80, -1, `v/10` ],
>                 [ 70, 5, `a/g` ] ] )
>   },
>   title=`Motion for Nagle/Saff parameters` );
```



This plot supplies good first approximations to a number of interesting characteristics of the jump. When the ripcord is pulled at the end of the first minute, the skydiver is still 1300 m above ground level and descending at an almost constant velocity of 49 m/s. Within 5 seconds from the time the parachute opens the motion settles into an essentially steady descent at 7 m/s; the total length of the jump is very close to 4 minutes.

More precise answers can be obtained from an explicit solution to (1). The solution during the free-fall portion of the jump is:

```
> SOL1 := dsolve( subs(k=k1,SYS) union IC,
>                 { x(t), v(t) } ):
> V1 := subs( SOL1, v(t) );
> X1 := subs( SOL1, x(t) );
```

$$V1 := -\frac{m\,g}{k1} + \frac{m\,g\,e^{\left(-\frac{t\,k1}{m}\right)}}{k1}$$

$$X1 := \left(-m\,g\,t\,k1 + m^2\,g + x0\,k1^2 - m^2\,g\,e^{\left(-\frac{t\,k1}{m}\right)}\right) / k1^2$$

"Initial conditions" for the second stage of the jump are

chosen to ensure continuity of both x and v at the time of deployment:

```
> IC2 := { v(td) = subs( t=td, V1 ),
>          x(td) = subs( t=td, X1 ) };
```

$$IC2 := \left\{ \mathrm{v}(\,td\,) = -\frac{m\,g}{k1} + \frac{m\,g\,\mathrm{e}^{\left(-\frac{td\,k1}{m}\right)}}{k1}, \mathrm{x}(\,td\,) = \left(-m\,g\,td\,k1 + m^2\,g + x0\,k1^{\,2} - m^2\,g\,\mathrm{e}^{\left(-\frac{td\,k1}{m}\right)} \right) /k1^{\,2} \right\}$$

When the chute is deployed, the motion is found to be

```
> SOL2 := dsolve( subs(k=k2,SYS) union IC2,
>                 { x(t), v(t) } ):
> V2 := subs( SOL2, v(t) );
> X2 := subs( SOL2, x(t) );
```

$$V2 := -\frac{m\,g}{k2} + \frac{m\,g\,\left(k1 - k2 + \mathrm{e}^{\left(-\frac{td\,k1}{m}\right)}\,k2\right)\,\mathrm{e}^{\left(-\frac{t\,k2}{m}\right)}}{\mathrm{e}^{\left(-\frac{td\,k2}{m}\right)}\,k2\,k1}$$

$$X2 := \left(-m\,g\,t\,k2 + m^2\,g + k2\,\left(-m^2\,g\,k1 + m^2\,g\,k1\,\mathrm{e}^{\left(-\frac{td\,k1}{m}\right)} + m\,g\,k1^{\,2}\,td - k2\,m\,g\,td\,k1 + m^2\,g\,k2 + k2\,x0\,k1^{\,2} - k2\,m^2\,g\,\mathrm{e}^{\left(-\frac{td\,k1}{m}\right)}\right) /k1^{\,2} - m^2\,g\,\left(k1 - k2 + \mathrm{e}^{\left(-\frac{td\,k1}{m}\right)}\,k2\right)\,\mathrm{e}^{\left(-\frac{t\,k2}{m}\right)} \Big/ \left(\mathrm{e}^{\left(-\frac{td\,k2}{m}\right)}\,k1\right)\right) /k2^{\,2}$$

The complete solution, for all $t > 0$, can be written in terms of the Heaviside function as

```
> V := V1 + (V2-V1)*Heaviside(t-td);
> X := X1 + (X2-X1)*Heaviside(t-td):
```

$$V := -\frac{m\,g}{k1} + \frac{m\,g\,\mathrm{e}^{\left(-\frac{t\,k1}{m}\right)}}{k1} + \left(-\frac{m\,g}{k2} + \frac{m\,g\,\left(k1 - k2 + \mathrm{e}^{\left(-\frac{td\,k1}{m}\right)}\,k2\right)\,\mathrm{e}^{\left(-\frac{t\,k2}{m}\right)}}{\mathrm{e}^{\left(-\frac{td\,k2}{m}\right)}\,k2\,k1} + \frac{m\,g}{k1} - \frac{m\,g\,\mathrm{e}^{\left(-\frac{t\,k1}{m}\right)}}{k1} \right) \mathrm{Heaviside}(\,t - td\,)$$

At deployment, the position and velocity are

```
> subs( t=60, PARAM[ns], [ X, V ] );
> evalf( " );
```

$$\left[\frac{5209}{4} - \frac{981}{4}\,\mathrm{e}^{(-12)}, -\frac{981}{20} + \frac{981}{20}\,\mathrm{e}^{(-12)}\right]$$

$$[\,1302.248493, -49.04969863\,]$$

In the same way, the time of impact is found to be[2]

```
> t_impact[ns]:=solve( subs(PARAM[ns],X2)=0, t);
> 't_impact[ns]' = evalf(t_impact[ns]);
```

$$t\_impact_{ns} := \frac{5}{7}\,\mathrm{W}\left(-\frac{\mathrm{e}^{\left(42\,\mathrm{e}^{(-12)} - \frac{331759}{981}\right)}\,\left(-6 + 7\,\mathrm{e}^{(-12)}\right)}{\mathrm{e}^{(-84)}}\right) - 30\,\mathrm{e}^{(-12)} + \frac{1658795}{6867}$$

$$t\_impact_{ns} = 241.5601768$$

The corresponding position and velocity are

```
> subs( t=t_impact[ns], PARAM[ns], [ X, V ] ):
> evalf( " );
```

$$[\,.1\,10^{-117}, -7.007142857\,]$$

These results show good agreement with the graphical solution; how does this motion compare with real-life? Is the free-fall terminal velocity of 49 m/s (110 mph) reasonable? Is a landing velocity of 7 m/s (15.6 mph) survivable? According to [10], terminal free-fall velocity is approximately 53.6 m/s (120 mph) and the impact velocity should be comparable to a free-fall from a five foot (1.52 m) high wall. The free-fall solution can be used to determine the "maximum survivable" impact velocity. When the parameter values are

```
> PARAM[five]:=x0=152/100, m=75, k1=15, g=981/100:
```

the corresponding motion is

```
> X5 := subs( PARAM[five], X1 );
> V5 := subs( PARAM[five], V1 );
```

$$X5 := -\frac{981}{20}\,t + \frac{24677}{100} - \frac{981}{4}\,\mathrm{e}^{(-1/5\,t)}$$

$$V5 := -\frac{981}{20} + \frac{981}{20}\,\mathrm{e}^{(-1/5\,t)}$$

The duration and impact velocity of this jump are

```
> T5 := fsolve( X5=0, t, 0..infinity );
> V_max := evalf( subs( t=T5, V5 ) );
```

$$T5 := .5671998658$$

$$V\_max := -5.26023068$$

Thus, the 7 m/s landing velocity exceeds this threshold by 33%. Similar non-physical conclusions are present in each of the other problems summarized in Table 1.

The previous example illustrates the need for appropriate parameter values. But, the real problem is with the specific form of the model. The Air Force Academy Training Guide states [10, p. 22] that the "opening

[2] Note that W is the Lambert W function[2], *i.e.*, the branch of the function W defined by $W(x)e^{W(x)} = x$ that is analytic at 0.

shock" ($j(t_d)$, where $j = \frac{da}{dt}$ is the jerk) is a "heavy but smooth tug". The mathematical interpretation of this is that the acceleration should be (at least) C^1 for $t > 0$. Explicit expressions for the acceleration and jerk can be found directly from the equations of motion:

```
> A := subs( k=K[pw_const], rhs(ode(SYS,v)) );
> J := subs( 'diff(v(t),t)'='a(t)', diff(A,t) );
```

$$A := -g- \left(k1 + \left(k2 - k1 \right) \mathrm{Heaviside}\left(t - td \right) \right) \mathrm{v}\left(t \right) / m$$

$$J := -\frac{\left(k2 - k1 \right) \mathrm{Dirac}\left(t - td \right) \mathrm{v}\left(t \right)}{m} - \left(k1 + \left(k2 - k1 \right) \mathrm{Heaviside}\left(t - td \right) \right) \mathrm{a}\left(t \right) / m$$

Thus, the acceleration and jerk do not possess the necessary smoothness (unless $k_1 = k_2$, *i.e.*, the parachute does not open). Note that the jump in the jerk, *i.e.*,

$$[j(t_d)] := \lim_{t \to t_d^+} j(t) - \lim_{t \to t_d^-} j(t),$$

is not simply the coefficient of the Heaviside function since the jump in the acceleration must also be considered. The following procedure provides a simple means to obtain the jump in any expression at a specified time.

```
> jump := ( f, pt ) ->
>   factor( limit( f, pt, right )
>         - limit( f, pt, left ) ):
```

The two jumps are now found to be

```
> Ajump := jump(A, t=td );
> Jjump := jump(subs(a(t)=A,J), t=td );
```

$$Ajump := \frac{\mathrm{v}\left(td \right) \left(-k2 + k1 \right)}{m}$$

$$Jjump := -\left(-k2 + k1 \right) \left(k1 \, \mathrm{v}\left(td \right) + m \, g + k2 \, \mathrm{v}\left(td \right) \right) / m^2$$

Using the Nagle/Saff parameter values, the jump in the acceleration is consistent with the graphical solution:

```
> subs( v(td)=subs(t=td,V), PARAM[ns], Ajump/g ):
> 'Ajump' = "*G, `` =evalf(",6)*G;
```

$$Ajump = \left(6 - 6 \, \mathrm{e}^{(-12)} \right) G, = 5.99996 \, G$$

```
> subs( v(td)=subs(t=td,V), PARAM[ns], Jjump/g ):
> 'Jjump' = " * `G/s`, `` = evalf(",6) * `G/s`;
```

$$Jjump = \left(\frac{48}{5} \, \mathrm{e}^{(-12)} - \frac{42}{5} \right) \, G/s, = -8.39994 \, G/s$$

One way to overcome the deficiencies in the original model is to generalize the form of the coefficient of air resistance. The model proposed in [6] (see also [8]) adds a transition layer to the coefficient of air resistance during which the air resistance will increase from the free-fall value to the final descent value. That is,

$$k(t) = \begin{cases} k_1, & t < t_d \\ k_d, & t_d \le t < t_d + \tau \\ k_2, & t \ge t_d + \tau \end{cases} \qquad (3)$$

where k_d is the coefficient of air resistance during deployment and τ is the length of time required for the chute to be fully deployed. Or, in Maple,

```
> Kgen := k1 + (KD-k1)*Heaviside(t-td)
>           + (k2-KD)*Heaviside(t-td-tau):
```

Note that choosing k_d to be the linear interpolant results in an acceleration that is only continuous. One choice of k_d that makes the acceleration continuously differentiable (on $t > 0$) is the interpolating cubic polynomial whose derivative vanishes at $t = t_d$ and $t = t_d + \tau$. Maple definitions of these functions are obtained from the following commands:.

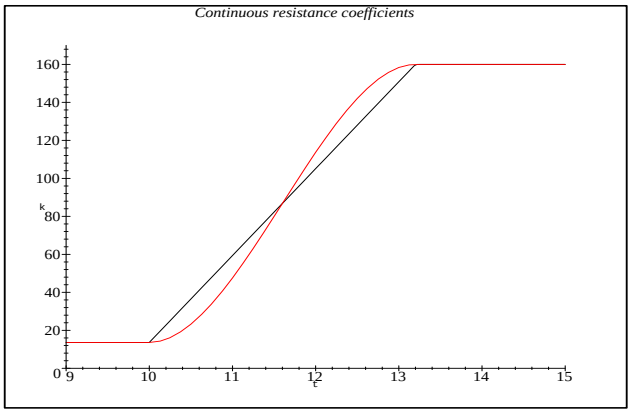
```
> Kd[linear] := alpha*t+beta:
> Kd[cubic] := int( alpha*(t-td)*(td+tau-t), t )
>            + beta:
> for nam in [ linear, cubic ] do
>   BC := Kd[nam]=kk;
>   solve( { subs(t=td, kk=k1, BC),
>            subs(t=td+tau, kk=k2, BC) },
>          { alpha, beta } );
>   DEPLOY := subs( ", Kd[nam] );
>   K[nam] := subs( KD=DEPLOY, Kgen );
> od:
```

The form of the coefficient of air resistance is only half of the story. It is also necessary to use realistic values of the parameter. Values consistent with the Air Force Academy Training Guide [10] are

```
> PARAM[af] := k1=2/11*75, k2=32/15*75, td=10,
>              tau=3.2, x0=4000*0.3048, m=75,
>              g=981/100:
```

With these constants, the coefficients of air resistance with the linear and cubic interpolants appear as

```
> plot(subs( PARAM[af], { K[linear], K[cubic] } ),
>      t=9..15, k=0..170,
>      title=`Continuous resistance coefficients`);
```



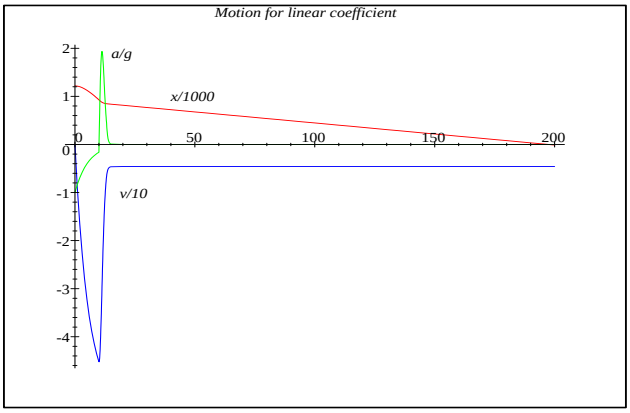
Explicit solutions for the linear and cubic coefficients can be found by a three-step process similar to the one

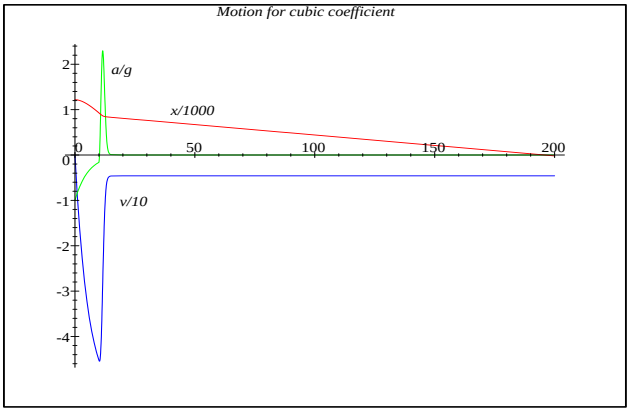
used previously for the piecewise constant coefficient. This process, and the accompanying results, are omitted due to the length and complexity of the solutions. Instead, each motion is computed using `motion` and displayed using `plot_motion`. The quality of the plot depends on the selection of an appropriate set of times. Since the jump height is significantly lower, the deployment and landing will occur earlier. The following set of times produces a good picture of the motion.

```
> snapshot[af] := [ $(0..8), i/10 $ i = 90 .. 149,
>                   5*i $ i = 3 .. 40 ]:
```

To see the effect of the smoothness of k on the jumps, the two motions are displayed using `plot_motion`:

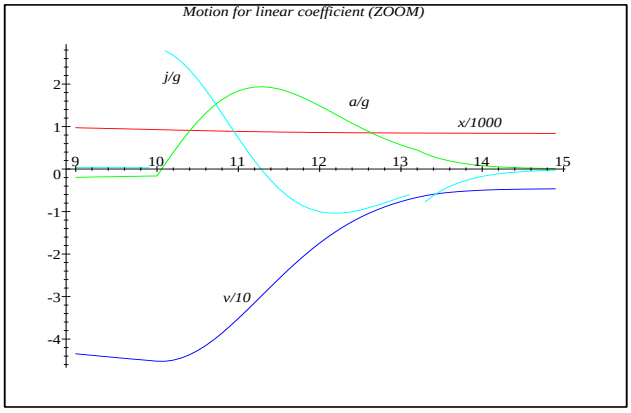
```
> for nam in [ linear, cubic ] do
>   IVP := subs( k=K[nam], SYS ) union IC;
>   PTS[nam] := motion( subs(PARAM[af],IVP),
>                       snapshot[af], t );
>   print( display(
>   {
>     plot_motion( PTS[nam], scale[noJ] ),
>     textplot( [ [ 50, 1, `x/1000` ],
>                 [ 25, -1, `v/10` ],
>                 [ 20, 1.9, `a/g` ] ] )
>   },
>   title=`Motion for `.nam.` coefficient` ) );
> od:
```

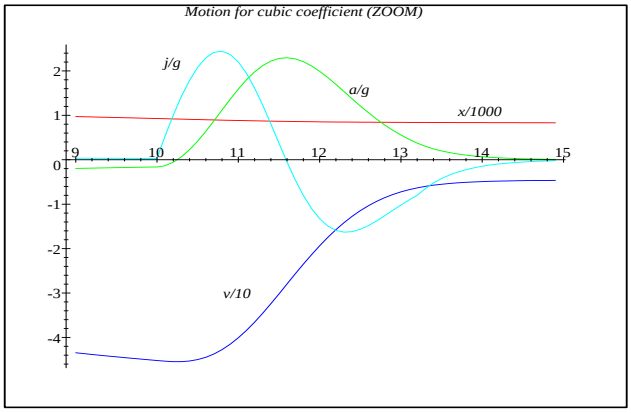




On this scale the motions are strikingly similar: impact occurs at ≈4.6 m/s after ≈196 s. This similarity is to be expected during the free-fall, and even in the final descent. Zooming in on the deployment stage of the jump, and including the jerk in the plot, provides a better look at the critical parts of the plots:

```
> scale[all] := { 'X'=1000, 'V'=10, 'A'=981/100,
>                 'J'=981/100 }:
> for nam in [ linear, cubic ] do
>   pts:=select( P->evalb(P[1]>=9 and P[1]<15),
>                PTS[nam] );
>   print( display(
>   {
>     plot_motion( pts, scale[all], heading ),
>     textplot( [ [ 14, 1.1, `x/1000` ],
>                 [ 11, -3, `v/10` ],
>                 [ 12.5, 1.6, `a/g` ],
>                 [ 10.2, 2.2, `j/g` ] ] )
>   },
>   title=`Motion for `.nam.
>         ` coefficient (ZOOM)` ) );
> od:
```





These plots illustrate several important qualitative differences between the two motions. In particular, note that while the acceleration is continuous for all $t > 0$ for both coefficients, the jerk is continuous only for the cubic interpolant. These issues are not solely of math-

ematical interest; these forces have a lot to say about the enjoyment – even survivability – of the jump. Additional analysis of these solutions can be found in [6].

Parameter Sensitivity

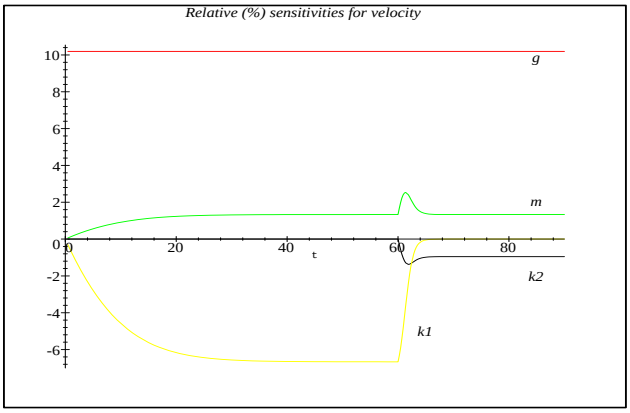
Specific parameter values must be selected for the preceding numerical and graphical analysis. However, these values are known with only a limited amount of certainty. For example, landing velocity should be between 15 and 17 ft/s (4.6 and 5.2 m/s) and the gravitational constant changes with altitude — particularly at altitudes above 25,000′ [11]. How dependent is the solution on each of the parameters?

One means of addressing this is to examine the relative rate of change of the position and/or velocity with respect to each of the parameters of the problem. (Since the only explicit solution available at this time is the solution with piecewise constant air resistance, this solution will be used to illustrate these ideas.) Since the position necessarily decreases to zero the relative sensitivities of the position can become unbounded. Thus, the remainder of this discussion focuses solely on the relative rates of change of the velocity with respect to the parameters k_1, k_2, g, and m. These expressions are easily computed from the explicit solution found earlier:

```
> Vrel := { seq( diff(V,p)/V*100,
>           p=[ k1, k2, g, m ] ) }:
```

While the jump lasts more than 4 minutes, the motion is essentially uniform after the first 90 seconds. Plotting the relative rates of change on this time interval yields:

```
> Vplot := plot( subs( PARAM[ns], Vrel ),
>                t=0..90 ):
> Vtext := textplot( [ [ 65, -5, 'k1' ],
>                      [ 85, -2, 'k2' ],
>                      [ 85, 9.8, 'g' ],
>                      [ 85, 2, 'm' ] ] ):
> display( { Vplot, Vtext },
>          title=`Relative (%) sensitivities`
>                .` for velocity` );
```



The general shape of these curves seem quite reasonable: the solution is a multiple of g, so the corresponding relative sensitivity is simply $1/g \approx 10.2\%$. The remaining curves show more temporal dependence, particularly before reaching either terminal velocity. The curves for both m and k_2 are asymptotic to the reciprocal of the parameter value ($1/m \approx 1.33\%$ and $1/k_2 \approx 0.95\%$). The k_1 curve approaches the corresponding asymptotic value, $1/k_1 \approx 6.7\%$, until the chute is deployed; after deployment, the value of this parameter quickly decays, as expected, to zero.

A Control Problem

Consider the question: What is the latest time the parachute can be deployed while keeping the landing velocity below 5.2 m/s (17 ft/s)? Assume that the C^1 air resistance is used and that all parameters (except deployment time) are taken from [10].

```
> PARAM[af2]:= k1 = 150/11, k2 = 160, tau = 32/10,
>              x0 = 6096/5, m = 75, g = 981/100;
> IVP:=subs(k=K[cubic], PARAM[af2], SYS union IC):
```

$$PARAM_{af2} := k1 = \frac{150}{11}, k2 = 160, \tau = \frac{16}{5}, x0 = \frac{6096}{5}, m = 75, g = \frac{981}{100}$$

There are several ways in which this problem can be solved; the approach used here is based on the numerical solution obtained from `motion`. The key to the efficient solution of this problem is an intelligent search algorithm. Mathematically, the problem can be posed as a boundary value problem that can be solved using, *e.g.*, a shooting method.[3] This is beyond the scope of the current discussion, so a hand search will be used.

To begin, note that the impact velocity for a jump according to the Air Force guidelines produced an impact velocity of 4.6 m/s. Thus, a lower bound is $t_d = 10$ s. A crude upper bound is the length of a jump without opening the chute (*i.e.*, $k \equiv k_1$):

```
> XX := subs( PARAM[af2], X1 ):
> VV := subs( PARAM[af2], V1 ):
> TT := fsolve( XX=0, t, 0..infinity );
```

$$TT := 28.06315629$$

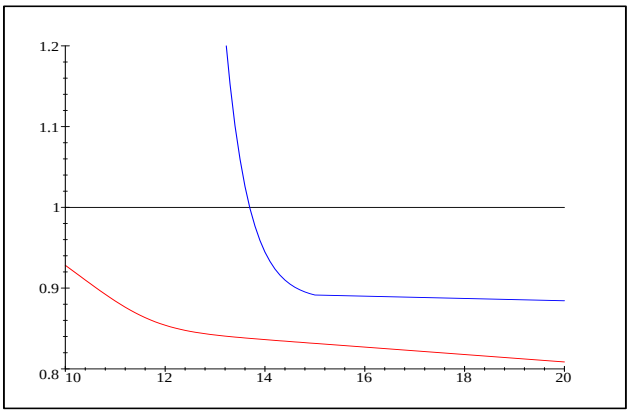
A binary search on the interval $[10, 28]$ using `motion` could find the optimal value of t_d to within 0.1 s in 8 iterations.

The fact that execution times for `motion` are approximately linear in the number of elements in the second argument make a binary search prohibitively slow. The search will be more efficient when the information in the solution is used in a more intelligent manner to guide

[3] see [7] for a Maple implementation of the shooting method

the iteration towards the optimal deployment time. For example, assuming the ripcord is pulled near terminal free-fall velocity, the skydiver falls approximately the same distance during the 3.2 seconds that it takes for the parachute to deploy; likewise, the rate of descent at the end of this period ($\approx$ 6.5 m/s) is essentially independent of t_d. The same logic applies to the final descent. How long does it take the speed to decrease from 6.5 m/s to the maximum safe value, 5.2 m/s? The resolution of the plots given earlier is not suitable for the accurate determination of this information. More accurate estimates can be obtained from `plot_motion` with intelligent scale factors

```
> display( { plot_motion( PTS[cubic],
>            { X=1000, V=-5.2 } ),
>            plot( 1, 10..20, 0.8..1.2 ) } );
```



Since $t_d = 10$ in this plot, it is seen that the velocity reaches the threshold in just over 3.7 s after the ripcord is pulled; the distance fallen in this period is approximately 90 m. This means that the largest value of t_d which ensures the landing velocity is below 5.2 m/s can be estimated by finding the free-fall time that puts the skydiver 90 m above the ground:

```
> fsolve( XX=90, t, 0..infinity );
```

$$26.38314907$$

The optimal value of t_d is expected to be less than this estimate. This information is useful in the determination of the points at which the motion should be computed.

```
> snapshot[af2] := [ 5*i$i=0..5,
>                    i/10 $ i = 260 .. 264,
>                    27, i/10 $ i=290..302 ]:
```

The motion for $t_d = 26.383$ s can now be found:

```
> pts := motion( subs( td=26.383, IVP ),
>                snapshot[af2], t ):
```

The analysis requires more detail than can be obtained from the plot (besides, plots consume too much space!). Instead, it suffices to extract the information for the times closest to $t = t_d$

```
> select( P->evalb(abs(P[1]-26.383)<0.1),
>         evalf(pts,5) );
```

$$[[\,26.300, 94.449, -53.503, -.082215, .014948\,],\\ [\,26.400, 89.098, -53.511, -.0718, 1.0477\,]\\ ]$$

and the times when the velocity is closest to the threshold

```
> select( P->evalb(abs(P[3]+5.2)<0.2),
>         evalf(pts,5) );
```

$$[[\,30., -12.318, -5.3923, 1.6936, -3.6130\,],[\\ 30.100, -12.850, -5.2398, 1.3682, -2.9189\\ ],[\\ 30.200, -13.367, -5.1166, 1.1054, -2.3581\\ ]]$$

Each data point contains five quantities: time, position, velocity, acceleration, and jerk. The velocity threshold is not met until the skydiver is below ground level; thus, as expected, this value for t_d is too large. To estimate an appropriate adjustment to t_d the acceleration value for $t = 30.1$ s can be used to extrapolate the velocity to find that the threshold velocity was met about 0.0291 s later and travelled another 0.1523 m. Thus, the ripcord should have been pulled when the parachute was 13.0015 m higher; the free-fall velocity at 26.3 s suggests that deployment should have begun 0.243 s earlier. This gives a second estimate of $t_d = 26.140$ s. The numerical solution to this problem is obtained and analyzed exactly as before:

```
> pts:=motion( subs(PARAM[af2], td=26.14, IVP),
>              snapshot[af2], t ):
> select( P->evalb(abs(P[1]-26.14)<0.1),
>         evalf(pts,5) );
> select( P->evalb(abs(P[3]+5.2)<0.2),
>         evalf(pts,5) );
```

$$[[\,26.100, 105.15, -53.486, -.085260, .015502\,],\\ [\,26.200, 99.799, -53.492, .0244, 3.5967\,]]$$

$$[[\,29.800, .48130, -5.3225, 1.5447, -3.2954\,],[\\ 29.900, -.043753, -5.1834, 1.2480,\\ -2.6623],\\ [\,30., -.55628, -5.0710, 1.0082, -2.1509\,]]$$

Extrapolating from the information at $t = 29.9$ s, the velocity threshold is crossed at $t = 29.887$ s with an extrapolated position of $x = 0.022$ m $= 2.2$ cm $< 1''$.

Thus, a jump with $t_d = 26.14$ s lands after 29.887 s with a landing velocity just below the 5.2 m/s threshold.[4] While it might be difficult to time the

[4] Of course, it must be remembered that these results are obtained by numerical integration of the equations of motion. Ad-

pulling of the ripcord this precisely and there is no time to consider using the reserve chute in an emergency, this does explain why some skydivers survive jumps when their chutes deploy only seconds before impact. In other words, it is almost never too late!

Conclusion

In this paper we have demonstrated the use of Maple in the analysis of a common textbook version of the parachute problem. Through a combination of graphic, numeric, and symbolic methods, the analysis identified several non-physical features of the original model and, subsequently, suggested the formulation of an improved model. While explicit solutions can be found for the new model, all analysis of the motions resulting from the improved model was either numerical or graphical. Many of the questions addressed in this paper — including parameter sensitivity and solution of a control problem — are not reasonable to attempt to solve using manual techniques. Maple, with its combination of symbolic, numeric, and graphic features, is the ideal software tool to use on these problems. The complete worksheet (Release 3 and Release 4) from which this article is produced is available from the author's WWW home page.

Acknowledgment

Financial support for this project has been received from NSF grants DMS-9404488 and OSR-9108722.

References

[1] M. L. Abell and J. P. Braselton, *Modern Differential Equations: Theory, Applications, Technology*, Harcourt Brace College Publishers, 1995.

[2] R. M. Corless, G. H. Gonnet, D. E. G. Hare, and D. J. Jeffrey, *Lambert's W function in Maple*, MapleTech, Issue 9, 1993, pp. 12–22.

[3] J. Drucker, *Minimal time of descent*, The College Mathematics Journal, 26 (1995), pp. 232–235.

[4] C. H. Edwards, Jr. and D. E. Penney, *Differential Equations and Boundary Value Problems: Computing and Modeling*, Prentice Hall, 1996.

[5] E. J. Kostelich and D. Armbruster, *Introductory Differential Equations: From Linearity to Chaos*, Addison–Wesley, 1996.

[6] D. B. Meade, *ODE models for the parachute problem*, SIAM Review, (submitted).

ditional tests should be conducted to ascertain the reliability and accuracy of these results.

[7] D. B. Meade, B. S. Haran, and R. E. White, *The shooting technique for the solution of two-point boundary value problems*, MapleTech, vol. 3, no. 1, 1996, pp. 85–93.

[8] R. Melka and D. Farrior, *Exploration of the parachute problem with STELLA*, Newsletter for the Consortium for Ordinary Differential Equations Experiments, Summer–Fall 1995, pp. 5–6.

[9] R. K. Nagle and E. B. Saff, *Fundamentals of Differential Equations*, Fourth Edition, Addison–Wesley, 1996.

[10] *Student Handbook for Airmanship 490: Basic Free Fall Parachuting*, USAF Academy, May 1990.

[11] H. S. Zim, *Parachutes*, Harcourt, Brace and Company, 1942.

Biography

Douglas B. Meade

Douglas B. Meade received a Ph.D. in Mathematics from Carnegie Mellon University in 1989, then spent two years as a Research Assistant Professor in the Center of Applied Mathematics at Purdue University. Since 1991 he has been an Assistant Professor in the Department of Mathematics at the University of South Carolina. Research interests are presently directed towards numerical methods for PDEs, in particular, wave propagation on unbounded domains and nonlinear reaction–diffusion equations. Dr. Meade has been active in the development of Maple resources for use in a variety of undergraduate and graduate courses. A Maple supplement to the Nagle/Saff ODE text was completed in 1996; a Maple module for the Engineer's Toolkit series, co-authored with Etan Bourkoff, will be published by Addison–Wesley in 1997.