Baseball Pitch Trajectory Project

Kyle McCotter & Luke Johnston

Biola University

Numerical Analysis

December 8, 2022

Abstract

This paper demonstrates the application and results of the Baseball Pitch Trajectory project for Math 321 - Numerical Analysis. This project applies the basic method published by Sportvision Randal Pendleton of Sportvision and forth-order Runge-Kutta in order to find the trajectory of a baseball pitch. The basic Sportsvision and forth-order Runge-Kutta methods will be compared to identify which is more accurate.

1 Introduction

In 2007 Sportsvision in association with Major League Baseball introduced the PITCHf/x system in order to get kinematic data on every pitch thrown in MLB ballparks (Kagan, 2009). This system uses two cameras in order to track the ball and then in turn calculate the position versus time. Sportsvision claims to have accuracy better than ½ inch to the exact location of the ball. In this paper we will explore two different methods of finding the trajectory given the data for two pitches.

The first step is to explore the simpler of the two methods known as the 9P or Sportvison model. This model takes nine inputs in order to calculate the position at a particular time. Next we will explore the second method which uses a fourth-order Runge-Kutta and in order to obtain a more accurate result we will use the Alan Nathan method to account for the drag and lift coefficients. At the end we will compare the results of these two methods to determine if the Runge-Kutta method is truly more accurate than the 9P model.

2 Sportvision Method

2.1 Theory

The Sportsvision method is the more straightforward method of the two and the code for it can be seen in Appendix A. The model uses the initial x, y, and z positions alongside the initial x, y, and z velocities and acceleration data in order to find the x, y, and z positions at a particular t. This is done for 50 iteterations at different values of t. The equation is given bellow.

$$\mathbf{x}_M(t) = \begin{pmatrix} X_o \\ Y_o \\ Z_o \end{pmatrix} + (t - t_o) \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} + \frac{1}{2}(t - t_o)^2 \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix}$$

2.1 Pitch #1

x	y	z	Vx	Vy	Vz	Ax	Ay	Az
-2.509	50	5.928	9.182	-132.785	-10.967	-19.268	30.713	-16.580

Table 1: Initial input values for pitch #1.

Final X	Final Y	Final Z
-0.3229	-11.3776	-1.4363

Table 2: Final position coordinates for pitch #1.

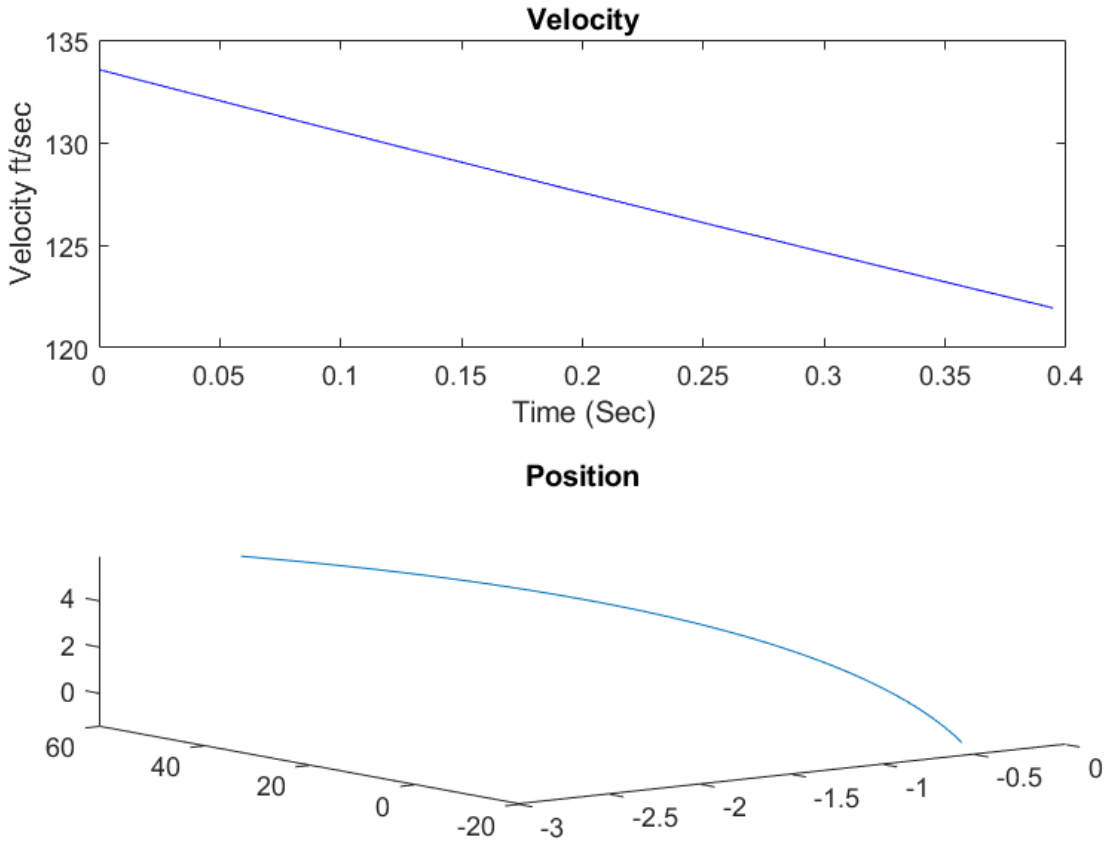

Figure 1: Graph for both the position and velocity of pitch #1.

2.1 Pitch #2

x	y	z	Vx	Vy	Vz	Ax	Ay	Az
-2.43	50	6.46	9.46	-143.17	-9.15	-23.08	34.2	-26.09

Table 3: Initial input values for pitch #1.

Final X	Final Y	Final Z
-0.565354	-16.047590	-1.155605

Table 4: Final position coordinates for pitch #1.

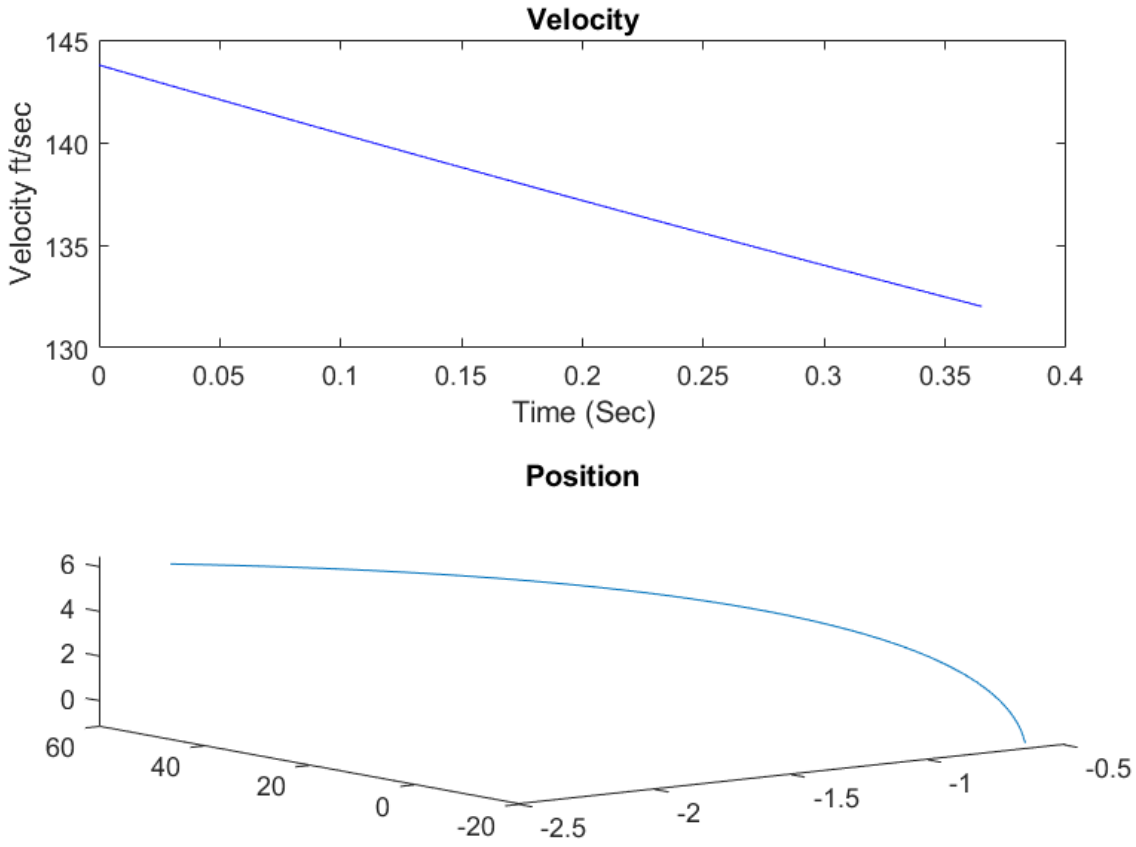

Figure 2: Graph for both the position and velocity of pitch #2.

3 Dr. Nathan's Method

3.1 Theory

Dr. Nathan claims that his method is a more accurate way to calculate baseball pitch trajectory. Nathan's method relies on a system of three equations, that find the values of x'', y'', and z''. Where x'' stands for the second derivative of x. x is the positon of the ball in the direction x at a given time t. x' (the first derivative of x) is the velocity in the of the ball in the

direction x at a given time. Using forth-order Runge-Kutta, with 100,000 iterations, we are able to solve the system of equations to find x', y', and z'. We can then use the first derivatives to find the values of x, y, and z. Below is the system of equations.

$$\begin{aligned}
\ddot{x} &= -KC_D v v_x - KC_L v v_y \sin\phi \\
\ddot{y} &= -KC_D v v_y + KC_L v \left(v_x \sin\phi - v_z \cos\phi\right) \\
\ddot{z} &= -KC_D v v_z + KC_L v v_y \cos\phi - g\,.
\end{aligned}$$

3.1 Pitch #1

x	**y**	**z**	**Vx**	**Vy**	**Vz**
-2.509	50	5.928	9.182	-132.785	-10.967
K	**Cd**	**Cl**	**ϕ**	**g**	
0.005152949	0.392648	0.255819	236.0038*pi/180	32.174	

Table 4: Initial input values for pitch #1.

Final X	**Final Y**	**Final Z**
-0.279956	-11.059959	-1.676028

Table 5: Final position coordinates for pitch #1.

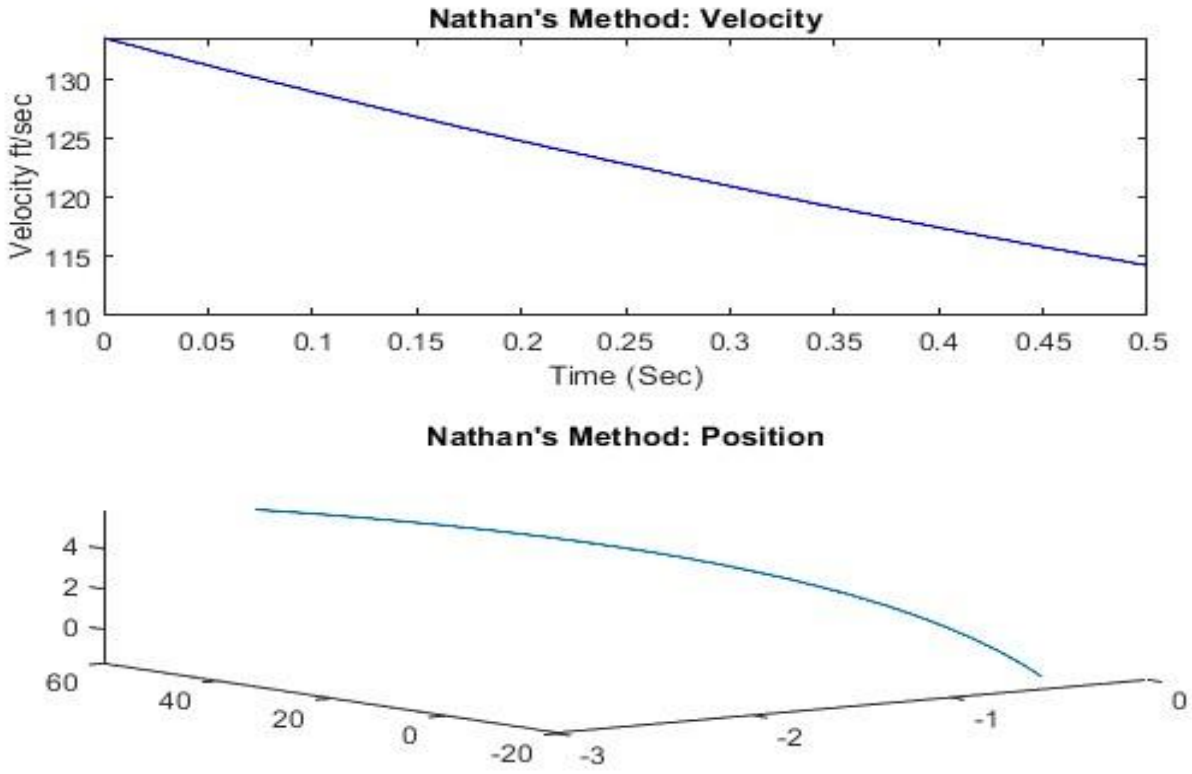

Figure 3: Graph for both the position and velocity of pitch #1.

3.1 Pitch #2

x	y	z	Vx	Vy	Vz
-2.43	50	6.46	9.46	-143.17	-9.5
K	**Cd**	**Cl**	**ϕ**	**g**	
0.005316103	0.392648	0.255819	4.591151161	32.174	

Table 6: Initial input values for pitch #2.

Final X	Final Y	Final Z
-0.934845	-14.901437	-1.460156

Table 7: Final position coordinates for pitch #2.

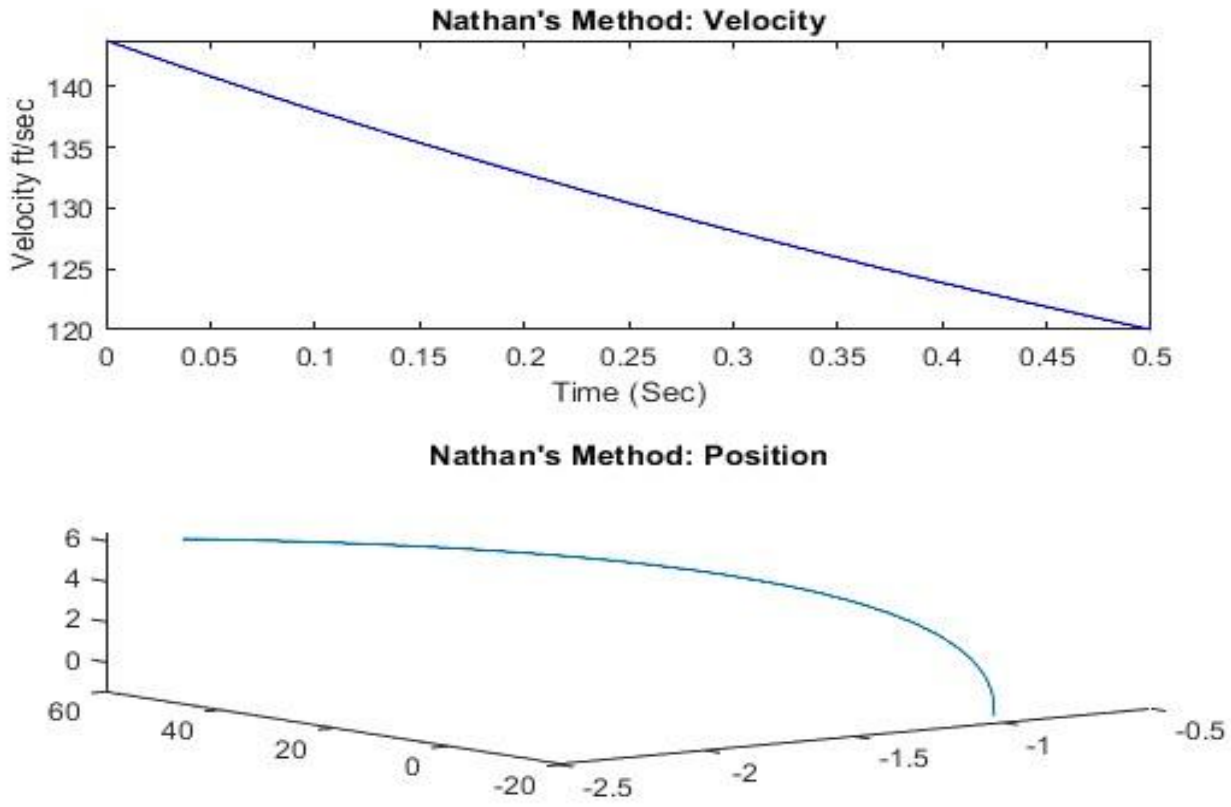

Figure 4: Graph for both the position and velocity of pitch #2.

4 Conclusion

We conclude that Nathan's method is more accurate than the Sportvision method. This increased accuracy is due to the second derivative system of equations that is solved using Runge-Kutta forth order. We are able to increase the accuracy by increasing the number of iterations used for the Runge-Kutta algorithm. Nathan's method used 100,000 iterations to find the solution, while Sportvision only used the recomended 50 iterations.

5 Appendix: Matlab Scripts

A.

```
% First Pitch
x = -2.509;
y = 50;
z = 5.928;

Vx = 9.182;
Vy = -132.785;
Vz = -10.967;

Ax = -19.268;
Ay = 30.713;
Az = -16.580;

K = 0.005152949;
Cd = 0.392648;
Cl = 0.255819;
phi = 236.0038*pi/180;
g = 32.174;

% Second Pitch
% x = -2.43;
% y = 50;
% z = 6.46;
%
% Vx = 9.46;
% Vy = -143.17;
% Vz = -9.5;
%
% Ax = -23.08;
% Ay = 34.2;
% Az = -26.09;
%
% K = 0.005316103;
% Cd = 0.392648;
% Cl = 0.255819;
% phi = 4.591151161;
% g = 32.174;

getTracjectory(x, y, z, Vx, Vy,Vz, Ax, Ay, Az)
rkAN(x, y, z, Vx, Vy, Vz, K, Cd, Cl, phi, g)

function getTracjectory(x, y, z, Vx, Vy,Vz, Ax, Ay, Az)
   n = 50;
   t = 0;
   xM = zeros(1,n);
```

```
    yM = zeros(1,n);
    zM = zeros(1,n);

    % Get Position
    for i=1:n
        xM(i) = x + (t*Vx) + (1/2)*(t^2)*Ax;
        yM(i) = y + (t*Vy) + (1/2)*(t^2)*Ay;
        zM(i) = z + (t*Vz) + (1/2)*(t^2)*Az;
        t = t+0.01;
        %fprintf("%f)\t%f \t %f \t %f\n", t, xM(i), yM(i), zM(i))
    end

    % Get Velocity
    accuray = 0.001;
    n2 = 50/accuray;
    vM = zeros(1, n2);
    tM = zeros(1, n2);
    i = 1;

    for y = 0 : 0.001 : 50
        [vM(i), tM(i)] = velocity(y, Vx, Vy,Vz, Ax, Ay, Az);
        i = i+1;
    end

    fprintf("Sportvison Method\nfinal positon values:\n      X          Y
Z\n")
    fprintf("%f\t%f \t %f \t %f \n", xM(n), yM(n), zM(n))

    tiledlayout(2,1)

    nexttile
    plot(tM, vM, 'b');
    xlabel('Time (Sec)');
    ylabel('Velocity ft/sec');
    title("Velocity")

    nexttile
    plot3(xM,yM,zM)
    title("Position")
end


function[v, t] = velocity(yAt, Vx, Vy, Vz, Ax, Ay, Az)
    % Get Velocity
    y0 = 50;
    t = (-Vy-sqrt(Vy^2-2*Ay*(y0-yAt)))/Ay;

    x = Vx + Ax*(t);
    z = Vz + Az*(t);
```

```
   y = Vy + Ay*(t);
   v = sqrt(x^2+z^2+y^2);
end


function rkAN(x, y, z, Vx, Vy,Vz, K, Cd, Cl, phi, g)

   % Setup Equations for ODE - This is the problem spot
   xEq = @(Vx, Vy, Vz) -K*Cd*(sqrt(Vx^2+Vy^2+Vz^2))*Vx -
K*Cl*(sqrt(Vx^2+Vy^2+Vz^2))*Vy*sin(phi);
   yEq = @(Vx, Vy, Vz) -K*Cd*(sqrt(Vx^2+Vy^2+Vz^2))*Vy +
K*Cl*(sqrt(Vx^2+Vy^2+Vz^2))*(Vx*sin(phi)-Vz*cos(phi));
   zEq = @(Vx, Vy, Vz) -K*Cd*(sqrt(Vx^2+Vy^2+Vz^2))*Vz +
K*Cl*(sqrt(Vx^2+Vy^2+Vz^2))*Vy*cos(phi) - g;

   n = 100000;
   h = (0.5)/n;

   % Setup Matrices
   xM = zeros(1, n);
   yM = zeros(1, n);
   zM = zeros(1, n);
   vM = zeros(1, n);
   vXM = zeros(1, n);
   vYM = zeros(1, n);
   vZM = zeros(1, n);
   tM = zeros(1, n);

   % Set Intial Values
   xM(1) = x;
   yM(1) = y;
   zM(1) = z;

   vM(1) = sqrt(Vx^2 + Vy^2 + Vz^2);
   vXM(1) = Vx;
   vYM(1) = Vy;
   vZM(1) = Vz;
   tM(1) = 0;

   fprintf("\n\nNathan's Method\n")
   %fprintf("\n\nPitch Position Data\n")
   %fprintf("%f)\t%f \t %f \t %f \t \t V=%f \n", 0, vXM(1), vYM(1),
vZM(1), vM(1))

   for i = 1:n
       % For X
       k1X = h * xEq(vXM(i), vYM(i), vZM(i));
       k2X = h * xEq(vXM(i) + k1X/2, vYM(i) + k1X/2, vZM(i) + k1X/2);
       k3X = h * xEq(vXM(i) + k2X/2, vYM(i) + k2X/2, vZM(i) + k2X/2);
```

```
        k4X = h * xEq(vXM(i) + k3X, vYM(i) + k3X, vZM(i) + k3X);
        vXM(i+1) = vXM(i) + (k1X + 2*k2X + 2*k3X + k4X) / 6;

        % For Y
        k1Y = h * yEq(vXM(i), vYM(i), vZM(i));
        k2Y = h * yEq(vYM(i) + k1Y/2, vYM(i)+ k1Y/2, vZM(i)+ k1Y/2);
        k3Y = h * yEq(vXM(i) + k2Y/2, vYM(i) + k2Y/2, vZM(i) + k2Y/2);
        k4Y = h * yEq(vXM(i) + k3Y, vYM(i) + k3Y, vZM(i) + k3Y);
        vYM(i+1) = vYM(i) + (k1Y + 2*k2Y + 2*k3Y + k4Y) / 6;

        % For Z
        k1Z = h * zEq(vXM(i), vYM(i), vZM(i));
        k2Z = h * zEq(vXM(i) + k1Z/2, vYM(i)+ k1Z/2, vZM(i)+ k1Z/2);
        k3Z = h * zEq(vXM(i) + k2Z/2, vYM(i) + k2Z/2, vZM(i) + k2Z/2);
        k4Z = h * zEq(vXM(i) + k3Z, vYM(i) + k3Z, vZM(i) + k3Z);
        vZM(i+1) = vZM(i) + (k1Z + 2*k2Z + 2*k3Z + k4Z) / 6;

        % Update Position
        xM(i+1) = xM(i) + vXM(i) * h;
        yM(i+1) = yM(i) + vYM(i) * h;
        zM(i+1) = zM(i) + vZM(i) * h;

        % Keep Track of Time and Velocity
        tM(i+1) = tM(i)+h;
        vM(i+1) = sqrt(vXM(i+1)^2+vYM(i+1)^2+vZM(i+1)^2);

        %fprintf("%f)\t%f \t %f \t %f \t \t V=%f \n", tM(i+1), xM(i+1),
yM(i+1), zM(i+1), vM(i+1))
    end

    fprintf("final positon values:\n      X          Y           Z
V\n")
    fprintf("%f\t%f \t %f \t %f \n", xM(n+1), yM(n+1), zM(n+1), vM(n+1))

    tiledlayout(2,1)

    nexttile
    plot(tM, vM, 'b');
    xlabel('Time (Sec)');
    ylabel('Velocity ft/sec');
    title("Nathan's Method: Velocity")

    nexttile
    plot3(xM,yM,zM)
    title("Nathan's Method: Position")
end
```

References

Cheney, W., Kincaid, D., (2013). Numerical Mathematics and Computing (7th ed.). Cengage Learning.

Kagan, David. “The Anatomy of a Pitch: Doing Physics with PITCHf/x Data.” The Physics Teacher 47 (October 2009): 412–16. http://baseball.physics.illinois.edu/KaganPitchfx.pdf.

Nathan, Alan. “Determining the Drag Coefficient from PITCHf/x Data” http://baseball.physics.illinois.edu/LiftDrag-1.pdf.