Simple interactive free throw optimizer

Paweł Brachaczek, 382318

Abstract

We devise a basketball free throw optimizer in Mathematica based on existing literature and our own simplifications. Taking into account player's height and other variables, such as basketball size, the program calculates the optimal release velocity and angle for a specified aim, e.g. towards the center of the hoop. We also illustrate the ball's trajectory on a graph. The results may be less accurate than those already existing in literature, since further optimization would significantly increase computation time and take away from "interactive experience".

Detailed explanation, theoretical background, etc.

Free throws often decide the outcome of basketball games. Players who struggle at the free throw line (converting less than 50% of their chances) can be fouled to the detriment of their team, lowering its expected value of points per possession. While there are many other explanations of why some high-profile players struggle shooting their free throws -- such as stress, fatigue, or incorrect shooting form -- optimizing the velocity and release angle can still increase a success probability and be of value to basketball teams.

In [1], Gablonsky and Lang propose a simple 2D model for free throws, ignoring the sideways error and spin of the ball. Using Newton's second law, they derive equations of motion in the horizontal and vertical directions, respectively:

$$m\,\frac{\mathrm{dv}_x}{\mathrm{dt}} = -\mathrm{kv}_x,$$
$$m\,\frac{\mathrm{dv}_y}{\mathrm{dt}} = \mathrm{mg} - \mathrm{kv}_y,$$

where m is mass of the ball; g -- standard gravity; k -- air viscosity coefficient. These equations can be analytically solved, yielding

$$\mathrm{x(t)} = \frac{\mathrm{mv}_o}{k}\cos(\theta_0)\,(1 - \exp[-\tfrac{k}{m}\,\mathrm{t}]),$$
$$\mathrm{y(t)} = \frac{\mathrm{mg}}{k}\,t - \frac{m}{k}\,(\mathrm{v}_0 \sin(\theta_0) - \frac{\mathrm{mg}}{k})\,\exp[-\tfrac{k}{m}\,\mathrm{t}] + \frac{m}{k}(\mathrm{v}_0 \sin(\theta_0) - \frac{\mathrm{mg}}{k}).$$

Values v_o and θ_0 correspond to release velocity and angle at release. We will want to find optimal values -- but we haven't defined yet what "optimal" means in our model.

First we define necessary conditions for successful trajectory. We assume the trajectory of the center of the ball starts at the point (1.25 x player's height, 0) -- since the player shoots from above their head [1] - and the position of the rim is defined in relation to this point, based on height of the rim (3.05 meters) and distance from the free throw line (about 4.6 meters, with small difference between NBA and FIBA rules). We also take small correction, since ball is usually released from ahead of the line (only player's feet need to be behind it). We can call trajectory "successful", if the ball passes above the rim (i.e. in moment *t* for which x(t) = distance to the front of the rim, y(t) > 3.05 + ball radius), and if later there is a moment *t* in which x(t) = aim and y(t) = 3.05. We can only aim towards the feasible points, such that the center of the ball is inside the hoop and the ball's distance from the rim is greater than the ball's radius. We do not account of the deflections, however we allow to aim towards the back of the rim -- such shot is successful as long as the center of the ball is not above the rim at the time of deflection (when x(t) = aim).

For specified aim, we can then find such v_o and θ_0 that maximize our objective function and satisfy necessary constraints (from the previous paragraph) up to a certain accuracy. Therefore we have to choose an objective function -- but because of interactive nature of our presentation, this function cannot be too costly to compute. In our model, we propose 4 different objectives:

1. Closest aim: we can choose such trajectory that the actual aim is as close as possible to the one we intended. Roughly, this option can reduce error from 10^{-4}m to 10^{-8}m, but since both of those numbers are very small, it's recommended to choose other options.
2. Minimal v_o (default): studies [2] find that players tend to choose trajectories with minimal release velocity, under the perception that minimizing used force also minimizes the error.
3. Minimal v_1, i.e. horizontal velocity when passing above the front of the rim. This objective is motivated by the conception that the "shooter's touch" requires minimal velocity as the ball approaches the rim, which also increases the chance of ball falling into the hoop even if trajectory is not successful by our assumptions (e.g. because of a "lucky" deflection). We define v_1 as x'(t).
4. Highest arc, measured when passing above the front of the rim. By common sense, this approach would yield the biggest absolute margin of error. Practicing such shots can also be useful for active-ball situations, when it's legal for another player to block a shot.

Further optimization, such as the one carried out in [1], would also require finding aim that has the biggest margin of error. To quantify both relative possible error in angle and velocity, we would need objective function that is perhaps suited particularly towards the struggles of player in need, e.g. placing 5 times as much weight on error in velocity. However, carrying out what is essentially optimization within optimization turned out to be too time-expensive.

In general, we would expect from our results to decrease in both optimal v_o and θ_0, as the player's height increases. While we do not compute a margin of error for trajectory, it's suggested [1] that the best aim is the one slightly towards the back of the rim (since hitting it is more "forgivable" that hitting the front of the rim). However, this trend might be less visible in our model, since we mostly do not account for deflections. Optimization method 2 roughly agrees with these preconceptions, and produce results similar to those from [1], especially in terms of velocity.

Model variables include:

- federation: FIBA (international organization) or NBA (North American league), influencing the distance between the rim and free throw line;

- ball size: 5, 6 or 7, influencing the ball’s mass and radius;
- city: Denver, Chicago or Seattle -- three cities with NBA history with differences in altitude and latitude, therefore influencing standard gravity *g;*
- temperature, at standard pressure the main factor behind the air viscosity *k;*
- player’s height, on which depends the point of release;
- aim, or rather how much do we deviate from the center of the rim.

We also include constants such as rim’s diameter and height (which do not vary between official competitions). All values, in metric system (which sometimes requires conversion ft -> m), are supplied in the code. Of model variables, only temperature has practically no impact on optimal trajectory (in realistic range).

Explanation of the code

The code is divided into two parts. The first part initializes all parameters and functions used in the second part, the interactive presentation.

First, we set physical parameters of a model. Most of them are functions of a single variable. Some take completely different values for different arguments, such as standard gravity *g* (with value -9.81 in general case, and more city-specific values for Denver, Chicago or Seattle) -- this is done by defining function several times for specific variables. Other functions, such as *airviscosity,* are just straightforward mathematical formulas, sourced in literature.

Then we define multivariate equations of horizontal and vertical motion, *x* and *y* respectively. These use parameters initialized in the previous part and correspond to theoretical formulas. We also use Mathematica functions, namely trigonometric and exponential functions.

Later we define 4 different optimization routines. All of them use the same variables, bar the first -- which specifies the routine selected. First 3 use NMinimize, the 4th -- NMaximize. They all optimize the same variables: *tmax* (which specifies the moment when the center of the ball goes through the hoop), *tmax2* (which specifies the moment when the center of the ball is directly above the front of the rim), *v0* and *angle* (which specify the velocity and angle on release). Two of the optimization constraints define moments *tmax* and *tmax2*, another ensures that $tmax > tmax2 > 0$. If not covered by the objective function, we also add constraints requiring that 1) at *tmax*, the ball is in desired position up to 0.1 mm; 2) at *tmax2,* the lowest point of the ball is above the rim. Since the routine nr 4, with the highest arc, tend to maximize velocity (sometimes to the point of being unrealistic), we put a limit on the *v0*. NMinimize and NMaximize also take starting intervals for variables, which are mostly based on experimentation and vary a bit between routines depending on expected outcome.

At last, we define graphical objects for later visualization. Ball is a disk (of proper size) centered on a point of release. Rim is a straight line in a proper distance from the point of release. *Inhoop* illustrates the moment when the ball goes through the hoop, this time looking from above and zoomed in 2x -- so the rim is a circle, and the ball, an orange disk, is centered depending on aim. Both *inhoop* and two texts below have set y coordinate, so they’re centered just below the rim. *Textv* displays velocity at release, while *texta* -- the release angle, converted from radians to degrees. We also draw an arrow from the point of release, illustrating a scaled vector of release velocity with release angle.

In the main program, we use Manipulate to display an interactive illustration of our results. The main part of it is ParametricPlot of solutions of equations x, y, with plugged in parameters depending on our choice and subsequent optimization (with results of optimization called by /.) We append the graphic objects independent of solutions as Prolog, and those dependent -- as Epilog. We also append *inhoop* graphic using Show, since it’s not a primitive. We set proper PlotRange and show only x axis due to aesthetic reasons. In order to ignore potential warnings which have minimal effect on the results, we put this procedure inside Quiet[].

We can manipulate variables such as player’s height, temperature, and aim, using slider or writing an exact value. Other parameters, namely federation, basketball size, city, and optimization routine, can be chosen from the list of radio buttons. Regardless of the names of variables in the code, we try to include user-friendly info and recommended default values. The last radio button bar, with optimization routines, is broken up into two columns to make a window more compact.

Code in Mathematica

Initialization

Physical parameters

```
In[•]:= g[city_] := -9.81;
        g["Denver"] := -9.798;
        g["Chicago"] := -9.804;
        g["Seattle"] := -9.811;

In[•]:= airviscosity[temperature_] := 2.791 * 10^(-7) * temperature^(0.7355);

In[•]:= balldiameter[7] := 0.76 / Pi;
        balldiameter[6] := 0.7225 / Pi;
        balldiameter[5] := 0.6925 / Pi;
        ballmass[7] := 0.6;
        ballmass[6] := 0.53;
        ballmass[5] := 0.48;

In[•]:= k[temperature_, size_] := 3 Pi * airviscosity[temperature] * balldiameter[size];

In[•]:= rimdiameter = 0.4545;
        rimheight = 3.05;

In[•]:= releaseheight[playerheight_] := 1.25 * playerheight;

In[•]:= releasedistance[federation_] := 5.8 - 1.575 - 0.0635;
        releasedistance["NBA"] := (14 - 3 / 12) * 0.3048 - 0.0635;
```

Equations and optimization methods

```
In[•]:= x[t_, temperature_, size_, v0_, angle_] := ballmass[size] v0 / k[temperature, size]
           Cos[angle] (1 - Exp[-k[temperature, size] / ballmass[size] t]);

In[•]:= y[t_, temperature_, size_, city_, v0_, angle_] :=
          ballmass[size] × g[city] / k[temperature, size] t - ballmass[size] / k[temperature, size]
            (v0 Sin[angle] - ballmass[size] × g[city] / k[temperature, size]) Exp[
             -k[temperature, size] / ballmass [size] t] + ballmass[size] / k[temperature, size]
            (v0 Sin[angle] - ballmass[size] × g[city] / k[temperature, size]);
```

```
In[ ]:= optimizationroutine["asclose", temperature_,
          size_, federation_, fromthemiddle_, city_, playerheight_] :=
         NMinimize[{Abs[x[tmax, temperature + 273.15, size, v0, angle] - releasedistance[
                federation] - fromthemiddle / 100 * (rimdiameter / 2 - balldiameter[size] / 2)],
            y[tmax, temperature + 273.15, size, city, v0, angle] ==
             rimheight - releaseheight[playerheight], x[tmax2, temperature + 273.15,
              size, v0, angle] == releasedistance[federation] - rimdiameter / 2,
            y[tmax2, temperature + 273.15, size, city, v0, angle] >
             rimheight - releaseheight[playerheight] + balldiameter[size] / 2, tmax > tmax2 > 0},
           {{tmax, 0.8, 1.2}, {tmax2, 0.75, 1.15}, {v0, 6, 8}, {angle, 50 Pi / 180, 60 Pi / 180}},
           MaxIterations → 100] [[2]];
        optimizationroutine["minv0", temperature_, size_, federation_,
          fromthemiddle_, city_, playerheight_] :=
         NMinimize[{v0, Abs[x[tmax, temperature + 273.15, size, v0, angle] - releasedistance[
                 federation] - fromthemiddle / 100 * (rimdiameter / 2 - balldiameter[size] / 2)] <
             10^(-4), y[tmax, temperature + 273.15, size, city, v0, angle] ==
             rimheight - releaseheight[playerheight], x[tmax2, temperature + 273.15,
              size, v0, angle] == releasedistance[federation] - rimdiameter / 2,
            y[tmax2, temperature + 273.15, size, city, v0, angle] >
             rimheight - releaseheight[playerheight] + balldiameter[size] / 2, tmax > tmax2 > 0},
           {{tmax, 0.8, 1.3}, {tmax2, 0.75, 1.25}, {v0, 6, 7.5}, {angle, 50 Pi / 180, 60 Pi / 180}},
           MaxIterations → 100] [[2]];
        optimizationroutine["minv1", temperature_, size_, federation_,
          fromthemiddle_, city_, playerheight_] :=
         NMinimize[{D[x[t, temperature + 273.15, size, v0, angle], t] /. t → tmax2,
            Abs[x[tmax, temperature + 273.15, size, v0, angle] - releasedistance[federation] -
               fromthemiddle / 100 * (rimdiameter / 2 - balldiameter[size] / 2)] < 10^(-4),
            y[tmax, temperature + 273.15, size, city, v0, angle] ==
             rimheight - releaseheight[playerheight], x[tmax2, temperature + 273.15,
              size, v0, angle] == releasedistance[federation] - rimdiameter / 2,
            y[tmax2, temperature + 273.15, size, city, v0, angle] >
             rimheight - releaseheight[playerheight] + balldiameter[size] / 2, tmax > tmax2 > 0},
           {{tmax, 0.8, 1.3}, {tmax2, 0.75, 1.25}, {v0, 6, 8}, {angle, 50 Pi / 180, 60 Pi / 180}},
           MaxIterations → 100] [[2]];
        optimizationroutine["highestarc", temperature_, size_, federation_,
          fromthemiddle_, city_, playerheight_] :=
         NMaximize[{y[tmax2, temperature + 273.15, size, city, v0, angle],
            Abs[x[tmax, temperature + 273.15, size, v0, angle] - releasedistance[federation] -
               fromthemiddle / 100 * (rimdiameter / 2 - balldiameter[size] / 2)] < 10^(-4),
            y[tmax, temperature + 273.15, size, city, v0, angle] == rimheight -
              releaseheight[playerheight], x[tmax2, temperature + 273.15, size, v0, angle] ==
             releasedistance[federation] - rimdiameter / 2, tmax > tmax2 > 0, v0 < 10},
           {{tmax, 0.8, 1.5}, {tmax2, 0.75, 1.45}, {v0, 6, 8}, {angle, 70 Pi / 180, 85 Pi / 180}},
           MaxIterations → 100] [[2]];
```

Graphic objects

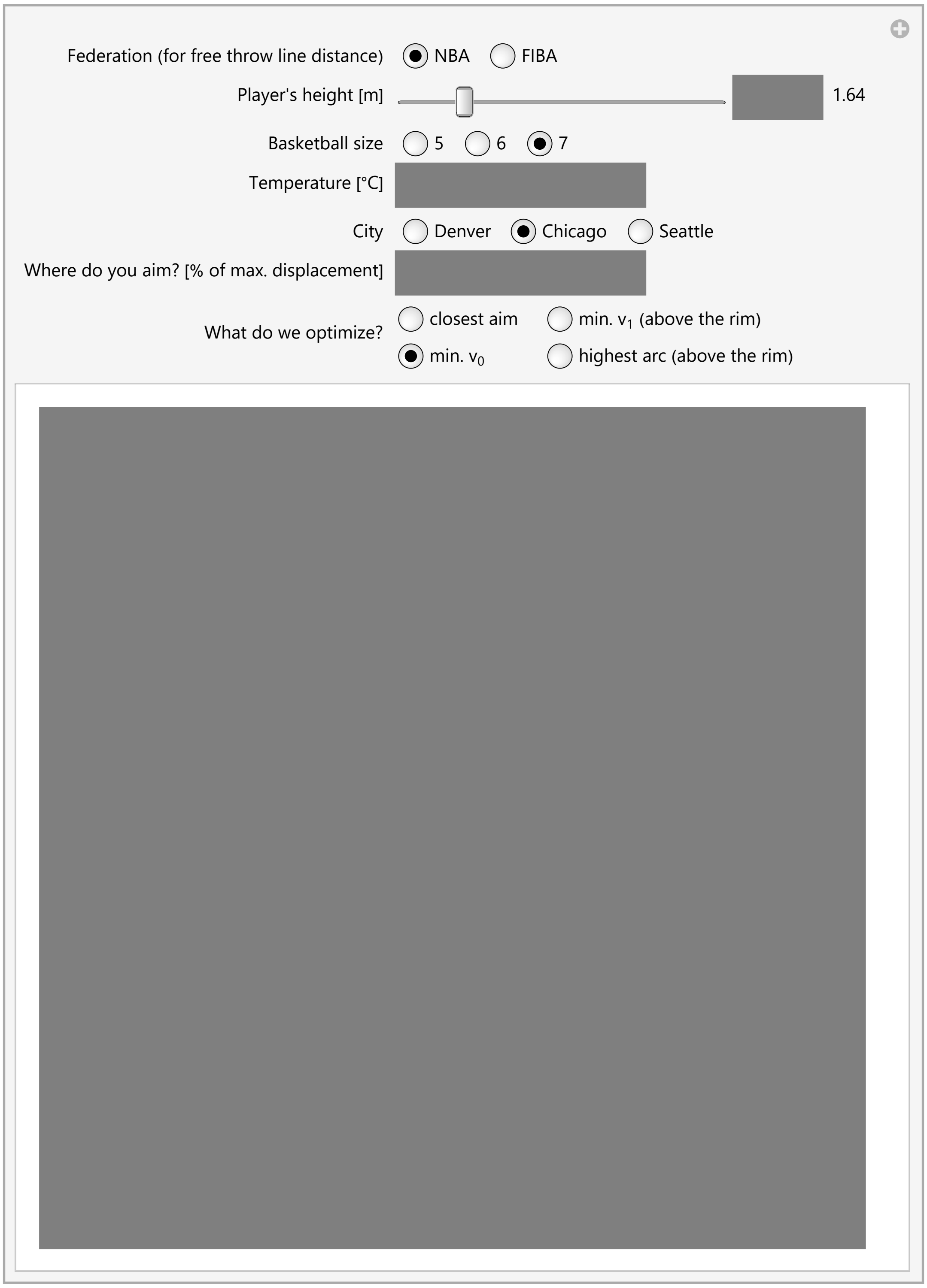
```
In[ ]:= ball[size_, playerheight_] :=
         Disk[{0, releaseheight[playerheight]}, balldiameter[size] / 2];

In[ ]:= rim[federation_] := Line[{{releasedistance[federation] - rimdiameter / 2, rimheight},
           {releasedistance[federation] + rimdiameter / 2, rimheight}}];
```

```
In[•]:= inhoop[fromthemiddle_, size_, federation_] :=
          Graphics[{Black, Circle[{releasedistance[federation], 2.3}, rimdiameter],
            Orange, Disk[{releasedistance[federation] + fromthemiddle / 50 *
                 (rimdiameter / 2 - balldiameter[size] / 2), 2.3}, balldiameter[size]]}];

In[•]:= textv[v0_, federation_] :=
          Text[Style[StringForm["v₀ = `1` m/s", NumberForm[v0, 4]], 14],
           {releasedistance[federation], 1.48}];
        texta[angle_, federation_] := Text[Style[StringForm["θ₀ = `1`°",
             NumberForm[angle * 180 / Pi, 4]], 14], {releasedistance[federation], 1}];

In[•]:= anglearrow[angle_, v0_, height_] := Arrow[{{0, releaseheight[height]},
            {0.05 * v0, releaseheight[height] + 0.05 * v0 * Tan[angle]}}];
```

Main program

```
In[•]:= Manipulate[Show[{Quiet[ParametricPlot[{x[t, temperature + 273.15, size, v0, angle],
              y[t, temperature + 273.15, size, city, v0, angle] + releaseheight[playerheight]},
             {t, 0, tmax}, PlotRange → {{-1, 5}, {0, 6}}, Axes → {True, False},
             Prolog → {rim[federation], Orange, ball[size, playerheight]},
             Epilog → {anglearrow[angle, v0, playerheight], textv[v0, federation],
               texta[angle, federation]}] /. optimizationroutine[routine,
             temperature, size, federation, fromthemiddle, city, playerheight]],
           inhoop[fromthemiddle, size, federation]}],
         {{federation, "NBA", "Federation (for free throw line distance)"},
          {"NBA", "FIBA"},
          ControlType → RadioButtonBar},
         {{playerheight, 1.95, "Player's height [m]"},
          1.5, 2.3, 0.01,
          Appearance → "Labeled"},
         {{size, 7, "Basketball size"}, {5, 6, 7},
          ControlType → RadioButtonBar},
         {{temperature, 22, "Temperature [°C]"}, 15,
          30, 1, Appearance → "Labeled"},
         {{city, "Chicago", "City"}, {"Denver", "Chicago", "Seattle"},
          ControlType → RadioButtonBar},
         {{fromthemiddle, 0, "Where do you aim? [% of max. displacement]"}, -100,
          100, 1, Appearance → "Labeled"}, {{routine, "minv0", "What do we optimize?"},
          {"asclose" → "closest aim", "minv0" → "min. v₀",
           "minv1" → "min. v₁ (above the rim)", "highestarc" → "highest arc (above the rim)"},
          ControlType → RadioButtonBar, Appearance → "Vertical" → {Automatic, 2}}]
```

Out[∘]=



Federation (for free throw line distance) [✓ NBA] FIBA

Player's height [m] 1.64

Basketball size 5 6 [✓ 7]

Temperature [°C]

City Denver [✓ Chicago] Seattle

Where do you aim? [% of max. displacement]

What do we optimize? closest aim; min. v_1 (above the rim); [✓ min. v_0]; highest arc (above the rim)

Literature

[1] Gablonsky J., Lang A. *Modelling basketball free throws.* SIAM Review 47/4, p. 775-798. https://digitalshowcase.oru.edu/cgi/viewcontent.cgi?article=1063&context=cose_pub

[2] Nakano N. et al. *Basketball players minimize the effect of motor noise by using near-minimum release speed in free-throw shooting.* Human Movement Science 70, 102583. https://www.-

sciencedirect.com/science/article/abs/pii/S0167945719307237
[3] http://www.fiba.basketball/documents/BasketballEquipment.pdf -- official FIBA rulebook with details on sizes of basketball equipment, namely the ball and the rim.
[4] https://en.wikipedia.org/wiki/Gravity_of_Earth#Comparative_values_worldwide -- table with values of standard gravity for different cities.
[5] https://upload.wikimedia.org/wikipedia/commons/6/6c/Basketball_courts.svg -- diagram showing dimensions of basketball court, including free throw line distance for both FIBA and NBA rulebook.
[6] https://www.tec-science.com/mechanics/gases-and-liquids/viscosity-of-liquids-and-gases/#Formulas_for_calculating_the_viscosity_of_air_and_water -- formula for air viscosity.