Quantitative Fundraising Appeal Prioritization System

written by Sean Madsen, IT manager for Bikes Not Bombs, 2013

Introduction

About this document

This document was made with *Mathematica* by Sean Madsen in 2013. You may be reading it in PDF form, which will help you get an understanding of the underlying algorithms used in the QFAP system. However, reading the .nb document with the *Mathematica* application will allow you interact and change it. The application is proprietary, so if you do not have it, you won't be able to modify the source document. But you do have the option of installing the free "Mathematica Reader" application to view the .nb file and interact with *some* of the graphs (but not make changes). If you will be reading this document in depth I highly recommend reading it using either *Mathematica* Reader, or *Mathematica.*

Intended Audience

The content herein serves to document the highly sophisticated process of culling contacts for fundraising appeals, as developed by Sean Madsen beginning in 2013. It is technical in nature and primarily designed for the computationally competent. However, concepts are first explained qualitatively so as to remain digestible for less mathematically inclined readers, should the need arise. It is not (yet?) designed for distribution outside of Bikes Not Bombs.

General overview of the process

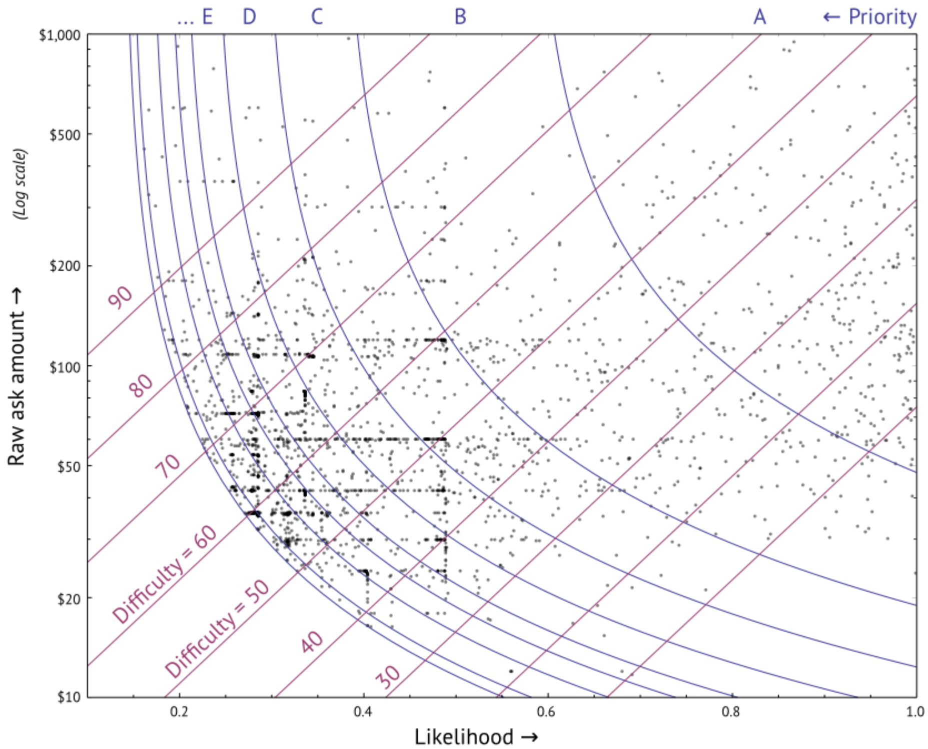
Bundling

First, all contacts are "bundled". Connections are made between contacts based on: (1) certain types of relationships ("household member", for example), and (2) presence of identical phone numbers. All connected contacts are "coagulated" into bundles. This means that if contact A is connected to contact B, and contact B is connected to contact C, then all three will end up in the same bundle with each other.

Measurements

After being bundled, we compute two important measurements for each bundle: the "**ask amount**" and the "**likelihood**". The ask amount is literally the amount of money we ask them to give. The likelihood is a number between 0 and 1 indicating how likely the bundle will be to donate.

After computing the ask amount and likelihood, we use those two values to calculate the "**priority**" and the "**difficulty**". Below is a graphic that reprsents these concepts visually:



Ask amount

Qualitative description

The **ask amount** is a dollar value that we would like to ask the donor to give.

We have two final values: the **raw ask amount** (which will look like $253.67) and the **sensible ask amount** (which will look like $250). The sensibility function converts raw ask amount values to sensible ask amount values in a smart way.

The raw ask amount is the product of 4 factors...

The mean factor: This value will be a dollar amount that reflects the central tendency of all contribution amounts. It is a weighted contraharmonic mean (a mean that will tend towards the higher values in a data set). The weights used are determined by the *full weight function*, which depends on how long ago the contribution was made (longer time ago means lower weight) and the contribution's simple weight (based on its type).

The slope factor: If the donor has been giving a consistent amount, the slope factor will be 1. If the amounts have been going up, the slope factor will be slightly more than 1, and if they've been going down, the slope factor will be slightly less than 1.

The active factor: If the donor has been giving recently, then the active factor will be 1. Otherwise it wll be slightly less than 1, meaning that if it’s been a while, we’ll do an easier ask.

The stretch factor: We also want to ask the donor for *more* than we think they’ll give. We stretch our ask according to the stretch function.

Constants (fixed values)

These values of these constants are fixed for the duraton of the search. The values are chosen through theoretical speculation, based on interacting with the Manipulate graphs below to produce sensible results.

flp	lehmer power (we use 2 here, for the contraharmonic mean)
fwdr	weight decay rate
fwds	weight decay sharpness
fsmx	maximum possible slope factor
fsmn	minimum possible slope factor
fsb	slope function base
fcps	points clout sharpness
fcpr	points clout rise

Variables

These values of these constants are fixed for the duraton of the search. The values are chosen through theoretical speculation, based on interacting with the Manipulate graphs below to produce sensible results.

per contribution

amt	amount
ybn	years before now
mul	multiplier
ws	weight, simple
wf	weight, full

per bundle

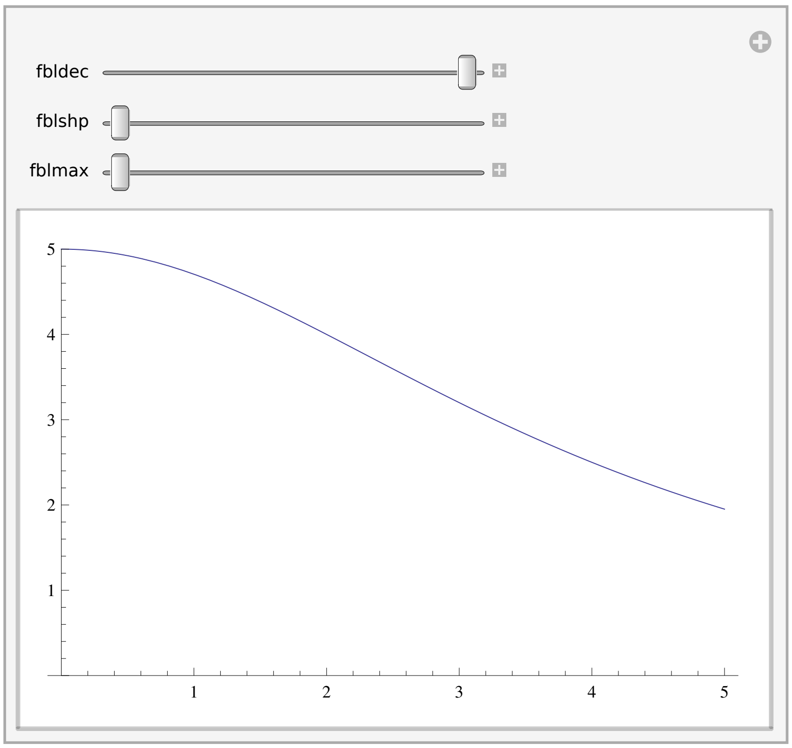
aar	raw ask amount
aas	sensible ask amount
num	number of contributions with wf > 0
clout	clout
zbsf	zero based slope factor
sf	slope factor

Functions

Baseline

$$\text{baselineFunc} := \text{baseline} :\to \frac{\text{fblmax}}{1 + (\text{yafl} / \text{fbldec})^{\text{fblshp}}};$$

```
Manipulate[Plot[baseline, {yafl, 0, 5}, PlotRange → {0, fblmax}],
  {{fbldec, 1.8}, 0, 4},
  {{fblshp, 3.3}, 2, 9},
  {{fblmax, 5}, 10, 100}] /. baselineFunc
```



Mean factor

Qualitative description

The mean factor is essentially a weighted contraharmonic mean of all contribution amounts, using complex weights. The weights are determined using the **full weight function.**

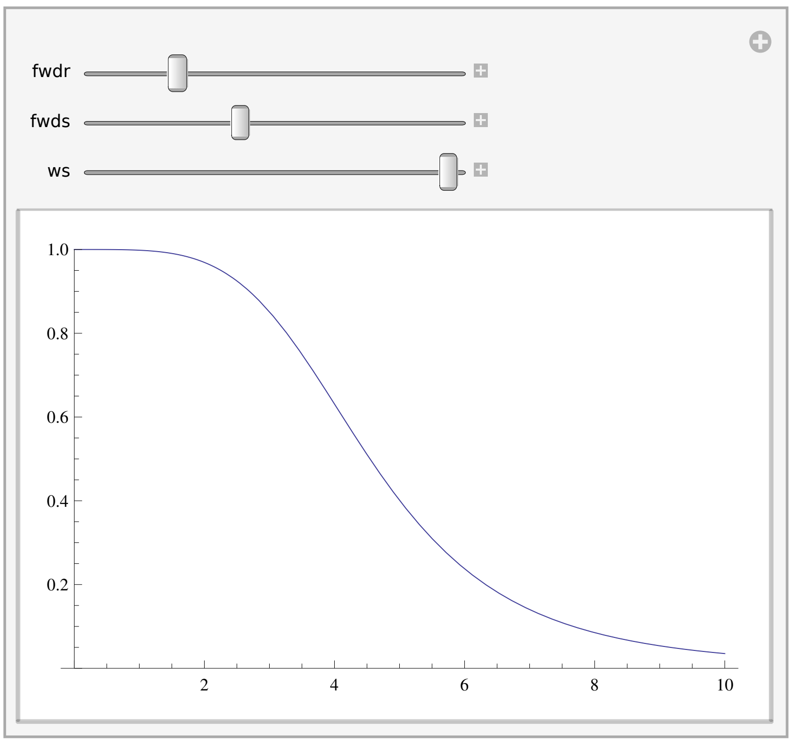
Calculation

$$\text{mean} = \frac{\text{Sum}\left[\text{wf} * (\text{amt} * \text{mul})^{\text{flp}}\right]}{\text{Sum}\left[\text{wf} * (\text{amt} * \text{mul})^{\text{flp}-1}\right]}$$

Full weight

The full weight is the weight value that we use for each contribution amount

```
wfFunc := wf :→ ws / (1 + (fwdr ybn)^fwds);
Manipulate[Plot[#, {ybn, 0, 10}, PlotRange → {0, 1}],
    {{fwdr, 0.22}, 0, 1},
    {{fwds, 4.2}, 1, 9},
    {{ws, 1}, 0, 1},
    SaveDefinitions → True] &@wf /. wfFunc
```



Rate (through regression analysis)

Old Initalizations

quadReg (old)

```
quadReg[points_, var_: x] :=
 Module[{n, x1, x2, y, w, x1t, x2t, x1x1t, x1x2t, x2x2t, yt, x1yt,
   x2yt, x1m, x2m, ym, sx1x1, sx1x2, sx2x2, sx1y, sx2y, b2, b3, b1},
  w = If[Length[points[[1]]] == 3, Sqrt[points[[All, 3]]], Table[1, {Length[points]}]];
  n = Length[points];
  x1 = points[[All, 1]] * w;
  x2 = (points[[All, 1]])^2 * w;
  y = points[[All, 2]] * w;
  x1t = Total[x1];
  x2t = Total[x2];
  yt = Total[y];
  x1x1t = Total[x1 * x1];
  x1x2t = Total[x1 * x2];
  x2x2t = Total[x2 * x2];
  x1yt = Total[x1 * y];
  x2yt = Total[x2 * y];
  x1m = x1t / n;
  x2m = x2t / n;
  ym = yt / n;
  sx1x1 = x1x1t - x1t^2/n;
  sx1x2 = x1x2t - (x1t * x2t)/n;
  sx2x2 = x2x2t - x2t^2/n;
  sx1y = x1yt - (yt * x1t)/n;
  sx2y = x2yt - (yt * x2t)/n;
  b2 = (sx1y * sx2x2 - sx2y * sx1x2)/(sx2x2 * sx1x1 - sx1x2^2);
  b3 = (sx2y * sx1x1 - sx1y * sx1x2)/(sx2x2 * sx1x1 - sx1x2^2);
  b1 = ym - b2 * x1m - b3 * x2m;
  b1 + b2 * var + b3 * var^2
 ]
```

quadReg

```
quadReg[points_, var_: x] := Module[{w, x1, x2, y, sx0x0, sx0x1,
    sx0x2, sx0y, sx1x1, sx1x2, sx1y, sx2x2, sx2y, det, b2, b3, b1},
  w = points[[All, 3]];
  x1 = points[[All, 1]];
  x2 = (points[[All, 1]])^2;
  y = points[[All, 2]];
  sx0x0 = Total[w];
  sx0x1 = Total[w * x1];
  sx0x2 = Total[w * x2];
  sx0y = Total[w * y];
  sx1x1 = Total[w * x1 * x1];
  sx1x2 = Total[w * x1 * x2];
  sx1y = Total[w * x1 * y];
  sx2x2 = Total[w * x2 * x2];
  sx2y = Total[w * x2 * y];
  det =
   - sx0x2^2 sx1x1 + 2 sx0x1 sx0x2 sx1x2 - sx0x0 sx1x2^2 - sx0x1^2 sx2x2 + sx0x0 sx1x1 sx2x2;
  b1 = (- sx0y sx1x2^2 + sx0x2 sx1x2 sx1y + sx0y sx1x1 sx2x2 -
      sx0x1 sx1y sx2x2 - sx0x2 sx1x1 sx2y + sx0x1 sx1x2 sx2y) / det;
  b2 = (sx0x2 sx0y sx1x2 - sx0x2^2 sx1y - sx0x1 sx0y sx2x2 + sx0x0 sx1y sx2x2 +
      sx0x1 sx0x2 sx2y - sx0x0 sx1x2 sx2y) / det;
  b3 = (- sx0x2 sx0y sx1x1 + sx0x1 sx0y sx1x2 + sx0x1 sx0x2 sx1y -
      sx0x0 sx1x2 sx1y - sx0x1^2 sx2y + sx0x0 sx1x1 sx2y) / det;
  b1 + b2 * var + b3 * var^2
 ]
```

reg

```
reg[points_, var_: x] := Module[{w, x1, x2, y, sx0x0, sx0x1, sx0x2, sx0y,
    sx1x1, sx1x2, sx1y, sx2x2, sx2y, det2, o1b1, o1b2, o2b2, o2b3, o2b1},
  w = points[[All, 3]];
  x1 = points[[All, 1]];
  x2 = (points[[All, 1]])^2;
  y = points[[All, 2]];
  sx0x0 = Total[w];
  sx0x1 = Total[w * x1];
  sx0x2 = Total[w * x2];
  sx0y = Total[w * y];
  sx1x1 = Total[w * x1 * x1];
  sx1x2 = Total[w * x1 * x2];
  sx1y = Total[w * x1 * y];
  sx2x2 = Total[w * x2 * x2];
  sx2y = Total[w * x2 * y];
  o1b1 = (sx0y sx1x1 - sx0x1 sx1y) / (- sx0x1^2 + sx0x0 sx1x1);
  o1b2 = (- sx0x1 sx0y + sx0x0 sx1y) / (- sx0x1^2 + sx0x0 sx1x1);
  det2 =
   - sx0x2^2 sx1x1 + 2 sx0x1 sx0x2 sx1x2 - sx0x0 sx1x2^2 - sx0x1^2 sx2x2 + sx0x0 sx1x1 sx2x2;
  o2b1 = (- sx0y sx1x2^2 + sx0x2 sx1x2 sx1y + sx0y sx1x1 sx2x2 -
      sx0x1 sx1y sx2x2 - sx0x2 sx1x1 sx2y + sx0x1 sx1x2 sx2y) / det2;
  o2b2 = (sx0x2 sx0y sx1x2 - sx0x2^2 sx1y - sx0x1 sx0y sx2x2 +
      sx0x0 sx1y sx2x2 + sx0x1 sx0x2 sx2y - sx0x0 sx1x2 sx2y) / det2;
  o2b3 = (- sx0x2 sx0y sx1x1 + sx0x1 sx0y sx1x2 + sx0x1 sx0x2 sx1y -
      sx0x0 sx1x2 sx1y - sx0x1^2 sx2y + sx0x0 sx1x1 sx2y) / det2;
  {o1b1 + o1b2 * var, o2b1 + o2b2 * var + o2b3 * var^2}
 ]
```

plotDonations zero based

```
contribs[id_] := SQLExecute[civi,
   "select yaf, rta, wf from contrib_worksheet where bundle_id = " ~~
    ToString@id ~~ " and cid is not null"];
```

```
plotDonations[contactID_] := Module[{id, a, now, wfit, xMin, xMax, yMin, yMax},
  a = contribs@contactID;
  id = ToString@contactID;
  wfit = quadReg@a;
  now = SQLExecute[civi, "select yaf from contrib_worksheet where bundle_id = " ~~
      id ~~ " and ybn = 0"][[1, 1]];
  Print[now];
  xMin = -0.05 * Max[a[[All, 1]]];
  xMax = now;
  yMin = 0;
  yMax = 1.3 * Max[a[[All, 2]]];
  Show[{ListPlot[a[[All, {1, 2}]], Joined → True, InterpolationOrder → 0,
      PlotStyle → {Gray, Dashed}], Plot[{wfit}, {x, xMin, xMax}, PlotStyle → Thick],
    Graphics[Red], Graphics[{Line[{{#1, #2}, {#1, #2 + #3 * 0.2 * yMax}}]} & @@@ a],
    Graphics[{Black, Text[Expand@wfit, {0.2 now, 0.8 yMax}]}]},
   PlotRange → {{xMin, xMax}, {yMin, yMax}}, AspectRatio → 1 / 3,
   ImageSize → 800, AxesOrigin → {now, 0}, PlotLabel → id]
 ]
```

Initalizations

reg

```
reg[points_, var_: x] := Module[{w, x1, x2, y, sx0x0, sx0x1, sx0x2, sx0y,
    sx1x1, sx1x2, sx1y, sx2x2, sx2y, det2, o1b1, o1b2, o2b2, o2b3, o2b1},
  w = points[[All, 3]];
  x1 = points[[All, 1]];
  x2 = (points[[All, 1]])^2;
  y = points[[All, 2]];
  sx0x0 = Total[w];
  sx0x1 = Total[w * x1];
  sx0x2 = Total[w * x2];
  sx0y = Total[w * y];
  sx1x1 = Total[w * x1 * x1];
  sx1x2 = Total[w * x1 * x2];
  sx1y = Total[w * x1 * y];
  sx2x2 = Total[w * x2 * x2];
  sx2y = Total[w * x2 * y];
  o1b1 = (sx0y sx1x1 - sx0x1 sx1y) / (- sx0x1^2 + sx0x0 sx1x1);
  o1b2 = (- sx0x1 sx0y + sx0x0 sx1y) / (- sx0x1^2 + sx0x0 sx1x1);
  det2 =
   - sx0x2^2 sx1x1 + 2 sx0x1 sx0x2 sx1x2 - sx0x0 sx1x2^2 - sx0x1^2 sx2x2 + sx0x0 sx1x1 sx2x2;
  o2b1 = (- sx0y sx1x2^2 + sx0x2 sx1x2 sx1y + sx0y sx1x1 sx2x2 -
      sx0x1 sx1y sx2x2 - sx0x2 sx1x1 sx2y + sx0x1 sx1x2 sx2y) / det2;
  o2b2 = (sx0x2 sx0y sx1x2 - sx0x2^2 sx1y - sx0x1 sx0y sx2x2 +
      sx0x0 sx1y sx2x2 + sx0x1 sx0x2 sx2y - sx0x0 sx1x2 sx2y) / det2;
  o2b3 = (- sx0x2 sx0y sx1x1 + sx0x1 sx0y sx1x2 + sx0x1 sx0x2 sx1y -
      sx0x0 sx1x2 sx1y - sx0x1^2 sx2y + sx0x0 sx1x1 sx2y) / det2;
  {o1b1 + o1b2 * var, o2b1 + o2b2 * var + o2b3 * var^2}
 ]
```

Database

```
Needs["DatabaseLink`"]
civi = OpenSQLConnection["local_bnb1civi"];
```

plotDonations negative

```
contribs[id_] := SQLExecute[civi,
   "select -ybn, rta, wf from contrib_worksheet where bundle_id = " ~~
    ToString@id ~~ " and cid is not null"];
```

```
plotDonations[contactID_] := Module[{id, a, now, wfit, xMin, xMax, yMin, yMax},
  a = contribs@contactID;
  id = ToString@contactID;
  wfit = reg@a;
  now = 0;
  xMin = 1.05 * Min[a[[All, 1]]];
  xMax = now;
  yMin = 0;
  yMax = 1.3 * Max[a[[All, 2]]];
  Show[{ListPlot[a[[All, {1, 2}]], Joined → True,
      InterpolationOrder → 0, PlotStyle → {Gray, Dashed}],
    Plot[wfit, {x, xMin, xMax}, PlotStyle → {Automatic, Thick}], Graphics[Red],
    Graphics[{Line[{{#1, #2}, {#1, #2 + #3 * 0.2 * yMax}}]} &@@@ a],
    Graphics[{Black, Text[Expand@wfit, {0.8 xMin, 0.8 yMax}]}]},
   PlotRange → {{xMin, xMax}, {yMin, yMax}}, AspectRatio → 1 / 3,
   ImageSize → 800, AxesOrigin → {now, 0}, PlotLabel → id]
 ]
```

Troubleshooting

```
contribs@27155
```

{{-6.5014, 500., 0.154664}, {-6.5014, 900., 0.0145566},
{-4.8712, 1400., 0.171134}, {-4.5479, 1900., 0.42452}, {-3.663, 2400., 0.60545},
{-2.1534, 2900., 0.814659}, {-1.2247, 3400., 0.846568}}

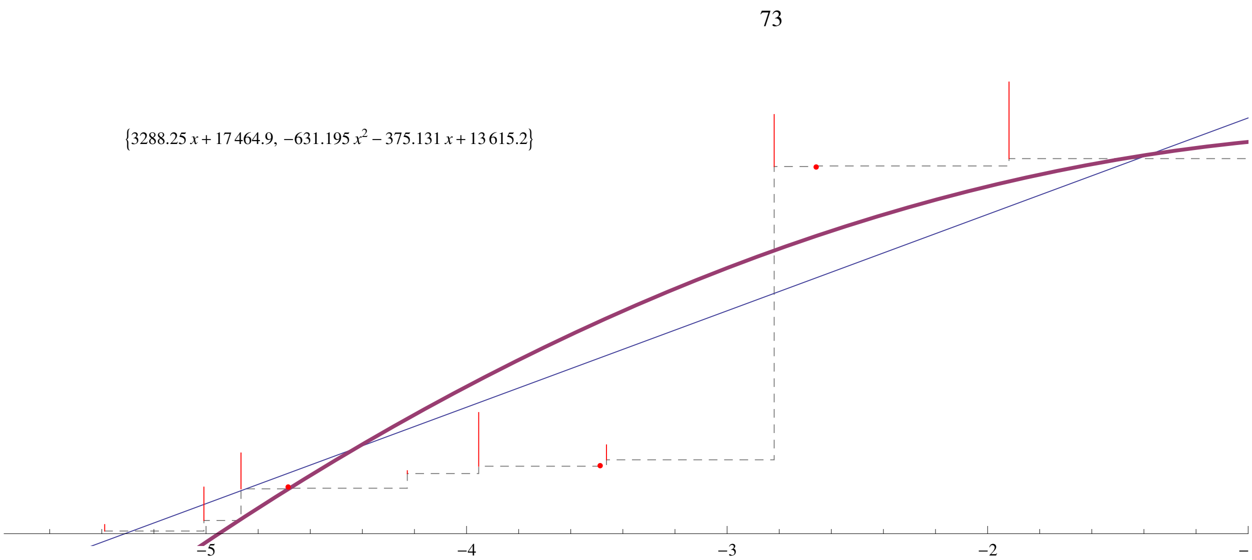
```
reg@contribs@27155
```

$\{4032.24 + 495.125\,x,\ 3708.24 + 247.322\,x - 36.6838\,x^2\}$

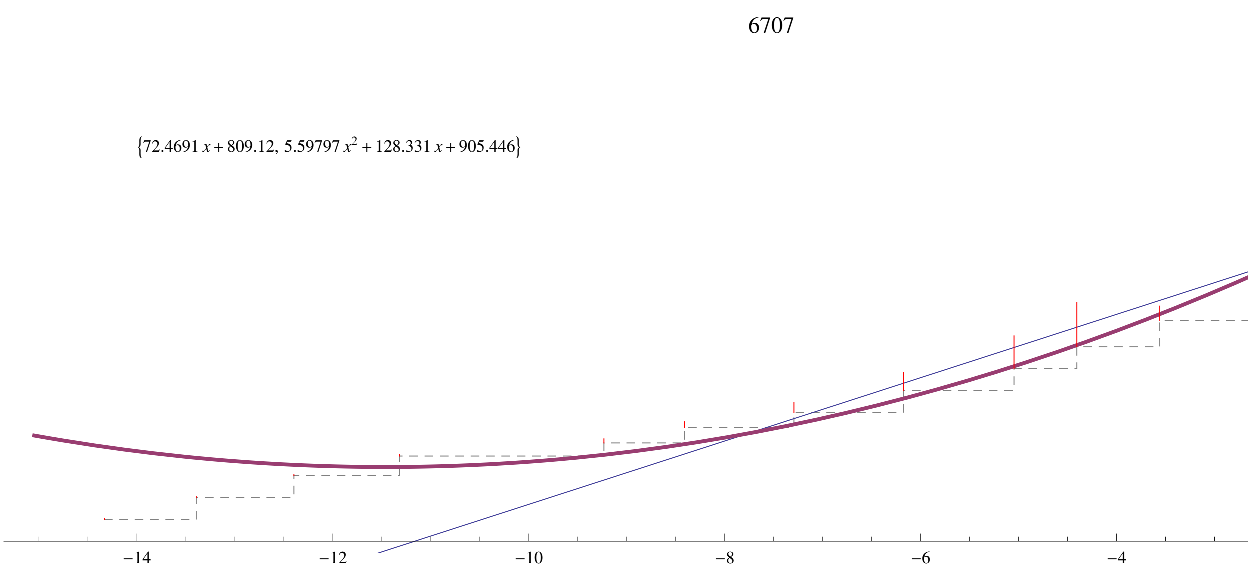
```
D[quadReg@contribs@27155, x] /. x → 3.5
```

500.552

```
plotDonations@73
```

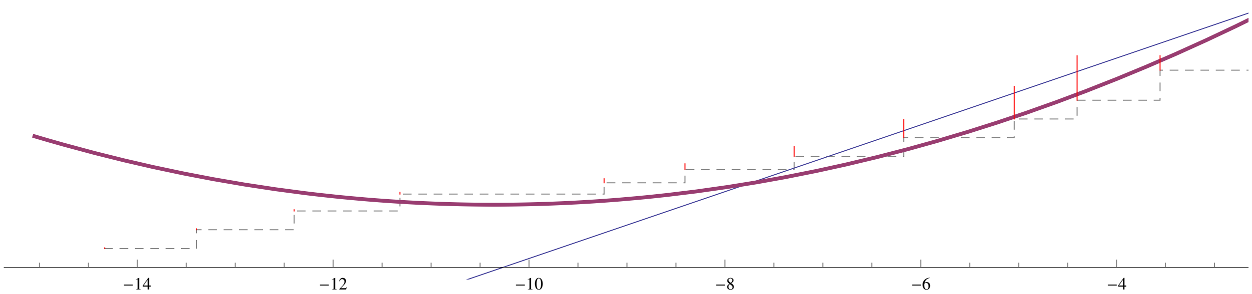



```
plotDonations@6707
```



6707

$\{88.9942\,x + 913.354,\ 8.28768\,x^2 + 171.696\,x + 1055.96\}$




```
points = contribs@73
```

```
{{0., 120., 0.0657001}, {0.3808, 480., 0.339649}, {0.5232, 1560., 0.364629},
 {0.7041, 1580., 0.}, {1.1616, 2080., 0.0287792}, {1.4356, 2330., 0.546103},
 {1.9013, 2370., 0.}, {1.926, 2610., 0.151623}, {2.5698, 12 610., 0.52887},
 {2.7315, 12 630., 0.}, {3.4712, 12 880., 0.779222}, {4.5013, 12 980., 0.998952}}
```

```
w = If[Length[points[[1]]] == 3, Sqrt[points[[All, 3]]], Table[1, {Length[points]}]]
```

```
{0.25632, 0.582794, 0.603845, 0., 0.169644,
 0.738988, 0., 0.389388, 0.727234, 0., 0.882735, 0.999476}
```

```
n = Length[points]
```

```
12
```

```
x1 = points[[All, 1]] * w
```

```
{0., 0.221928, 0.315932, 0., 0.197059,
 1.06089, 0., 0.749961, 1.86885, 0., 3.06415, 4.49894}
```

```
x2 = (points[[All, 1]])^2 * w
```

```
{0., 0.0845102, 0.165295, 0., 0.228904,
 1.52301, 0., 1.44442, 4.80256, 0., 10.6363, 20.2511}
```

```
y = points[[All, 2]] * w
```

```
{30.7584, 279.741, 941.998, 0., 352.86,
 1721.84, 0., 1016.3, 9170.43, 0., 11 369.6, 12 973.2}
```

```
x1t = Total[x1]
```

```
11.9777
```

```
x2t = Total[x2]
```

```
39.1361
```

```
yt = Total[y]
```

37 856.8

```
x1x1t = Total[x1 * x1]
```

34.9979

```
x1x2t = Total[x1 * x2]
```

135.49

```
x2x2t = Total[x2 * x2]
```

550.794

```
x1yt = Total[x1 * y]
```

113 360.

```
x2yt = Total[x2 * y]
```

432 044.

```
x1m = x1t / n
```

0.998142

```
x2m = x2t / n
```

3.26134

```
ym = yt / n
```

3154.73

$$sx1x1 = x1x1t - \frac{x1t^2}{n}$$

23.0424

$$sx1x2 = x1x2t - \frac{x1t * x2t}{n}$$

96.4266

$$sx2x2 = x2x2t - \frac{x2t^2}{n}$$

423.158

$$sx1y = x1yt - \frac{yt * x1t}{n}$$

75 573.7

$$sx2y = x2yt - \frac{yt * x2t}{n}$$

308 580.

$$b2 = \frac{sx1y * sx2x2 - sx2y * sx1x2}{sx2x2 * sx1x1 - sx1x2^2}$$

4915.4

$$b3 = \frac{sx2y * sx1x1 - sx1y * sx1x2}{sx2x2 * sx1x1 - sx1x2^2}$$

-390.859

```
b1 = ym - b2 * x1m - b3 * x2m
```

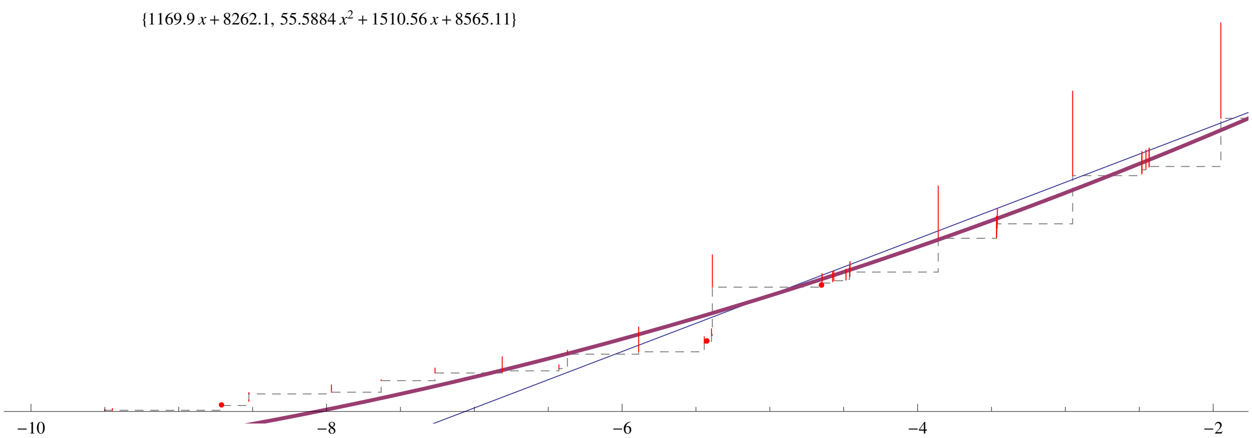
-476.818

Nice examples

```
examples = Sort@{39 036, 49 489, 27 050, 110,
   5999, 6707, 13 362, 4681, 11 207, 14 054, 27 155, 55 358, 35 761}
plotDonations /@ examples // TableForm
```

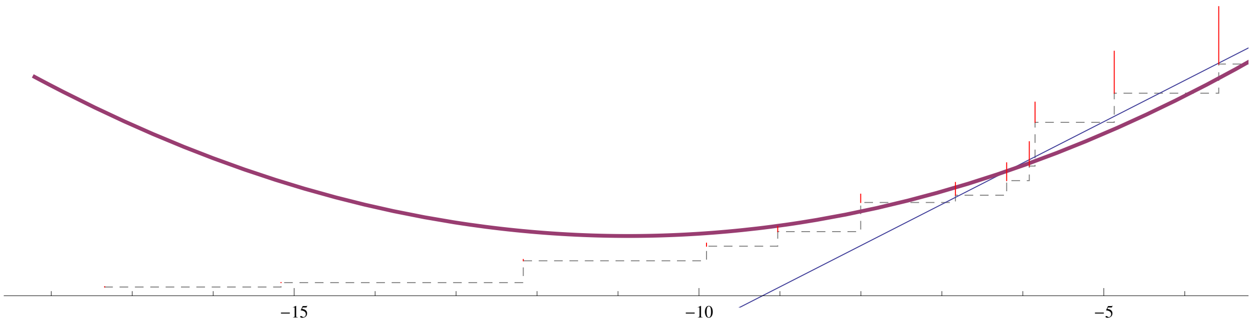
{110, 4681, 5999, 6707, 11 207, 13 362, 14 054, 27 050, 27 155, 39 036, 49 489}

110



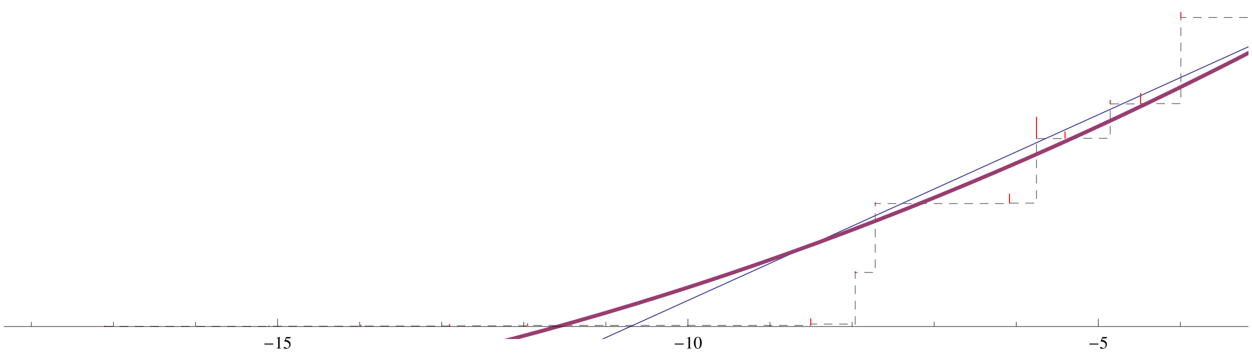
4681

$\{141.681\,x+1305.66,\ 10.1638\,x^2+221.093\,x+1407.44\}$



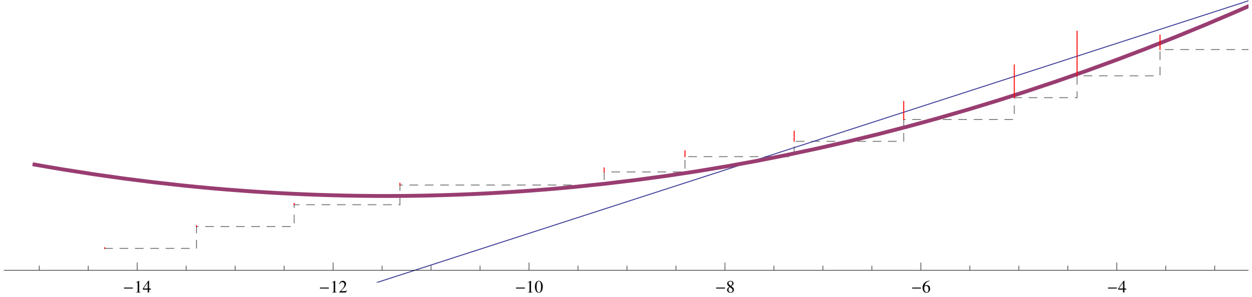
5999

$\{8627.89\,x+92\,338.6,\ 267.013\,x^2+11\,489.9\,x+97\,278.1\}$



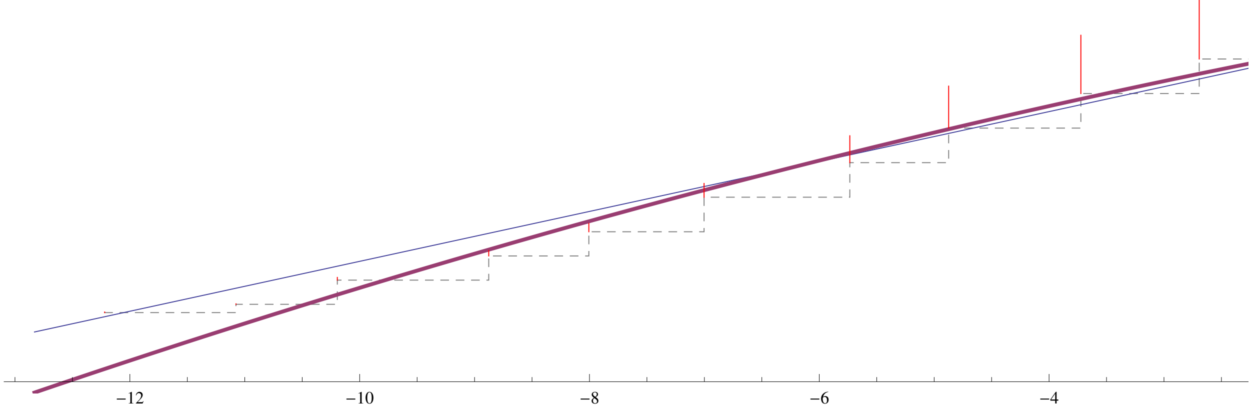
6707

$\{72.4691\,x+809.12,\ 5.59797\,x^2+128.331\,x+905.446\}$



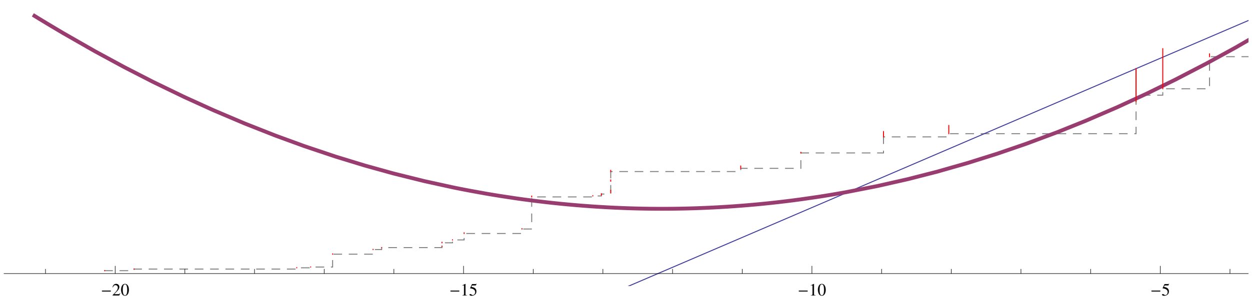
11207

$\{36.1134\,x + 535.317,\ -1.10087\,x^2 + 28.4226\,x + 530.16\}$



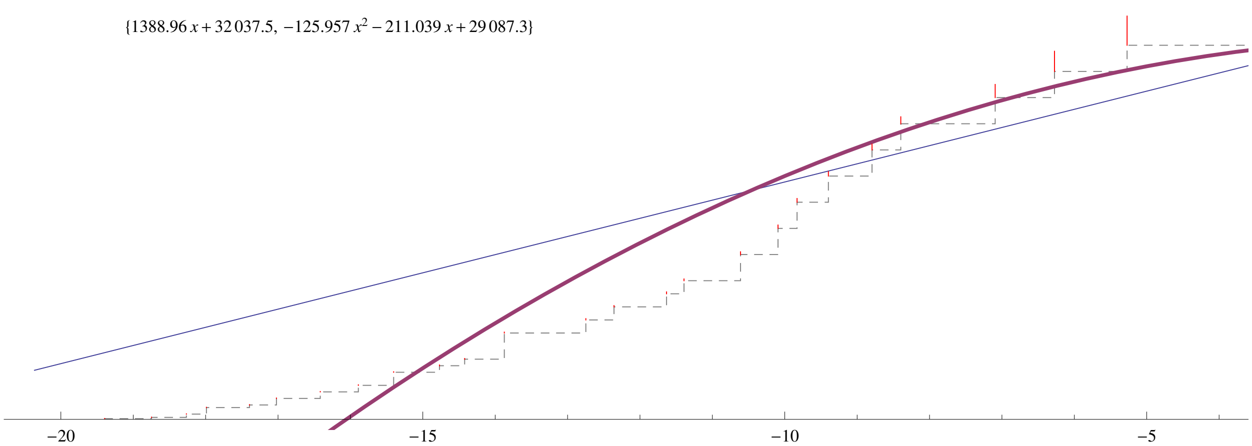
13362

$\{465.244\,x + 5680.79,\ 37.0601\,x^2 + 900.744\,x + 6482.54\}$



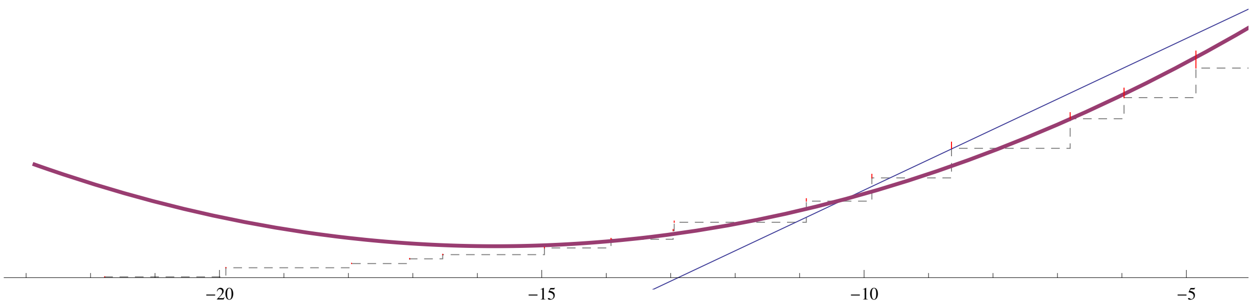
14054

$\{1388.96\,x + 32\,037.5,\ -125.957\,x^2 - 211.039\,x + 29\,087.3\}$



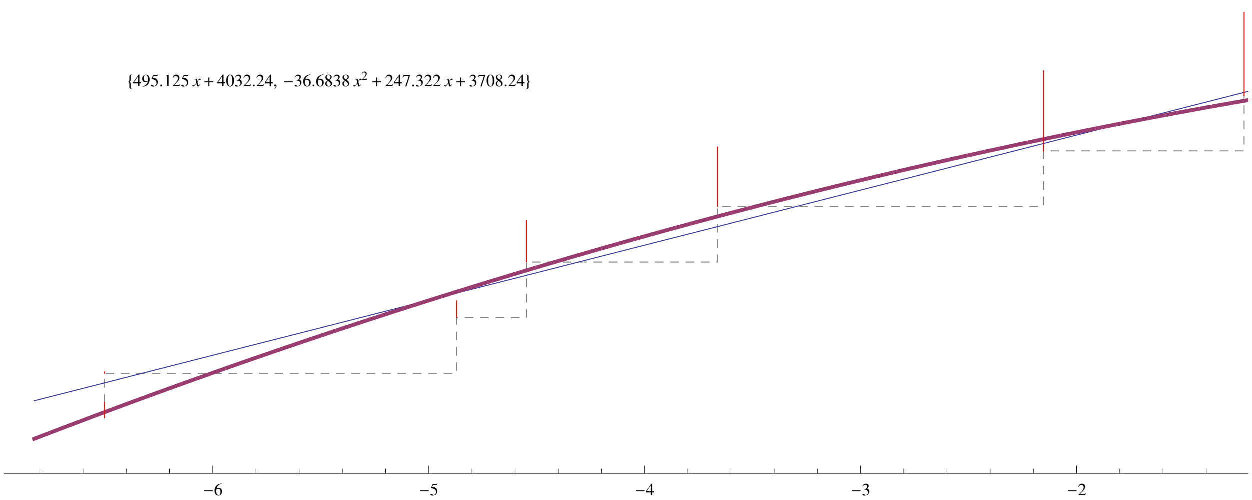
27050

$\{3590.84\,x + 46\,268.7,\ 189.241\,x^2 + 5949.94\,x + 50\,489.8\}$



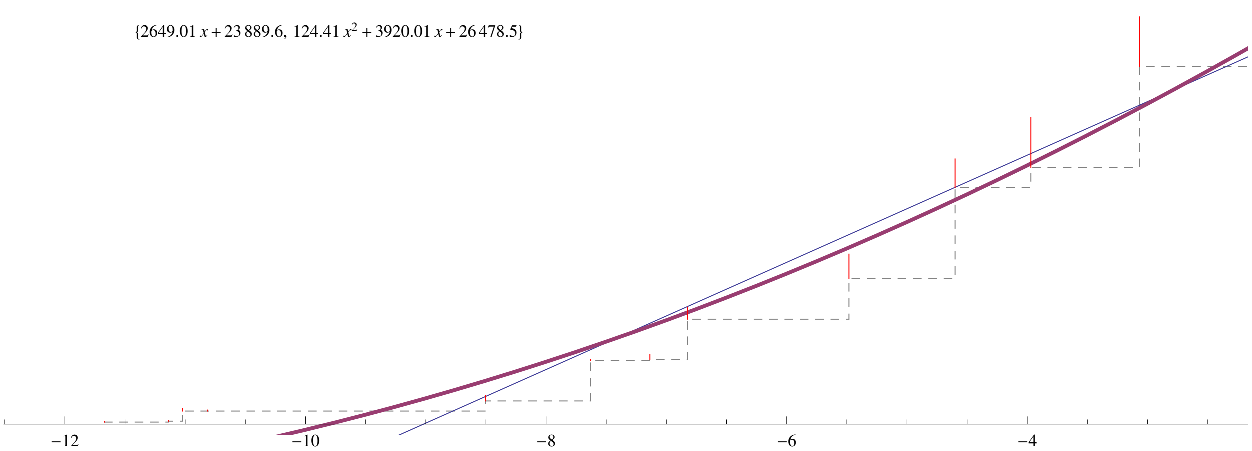
27155

$\{495.125\,x + 4032.24,\ -36.6838\,x^2 + 247.322\,x + 3708.24\}$

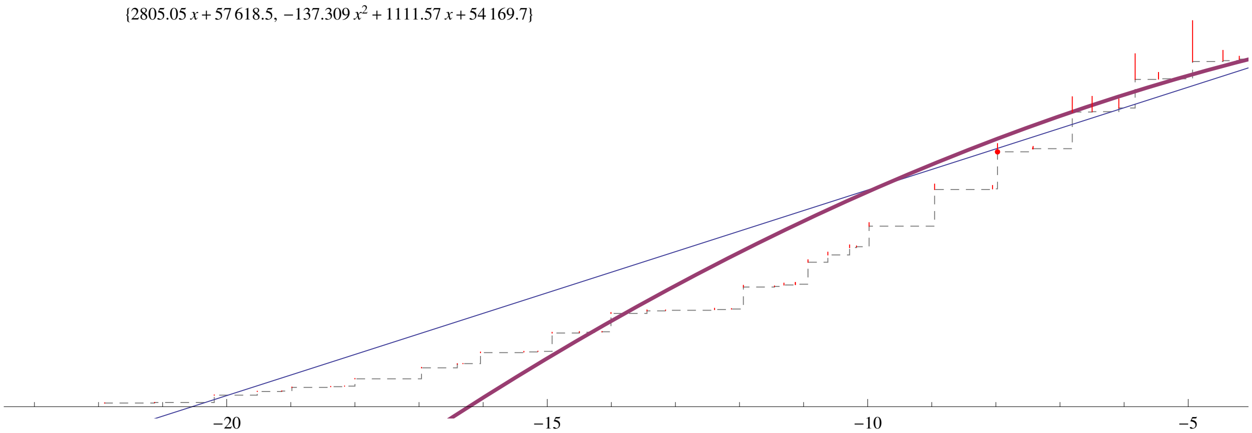


39036

$\{2649.01\,x + 23\,889.6,\ 124.41\,x^2 + 3920.01\,x + 26\,478.5\}$



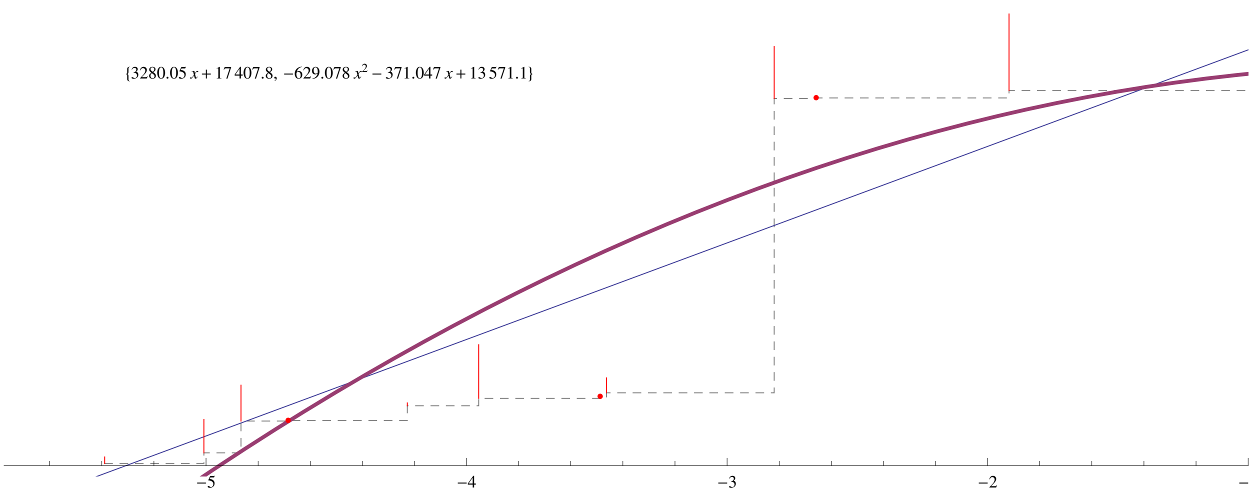
49489



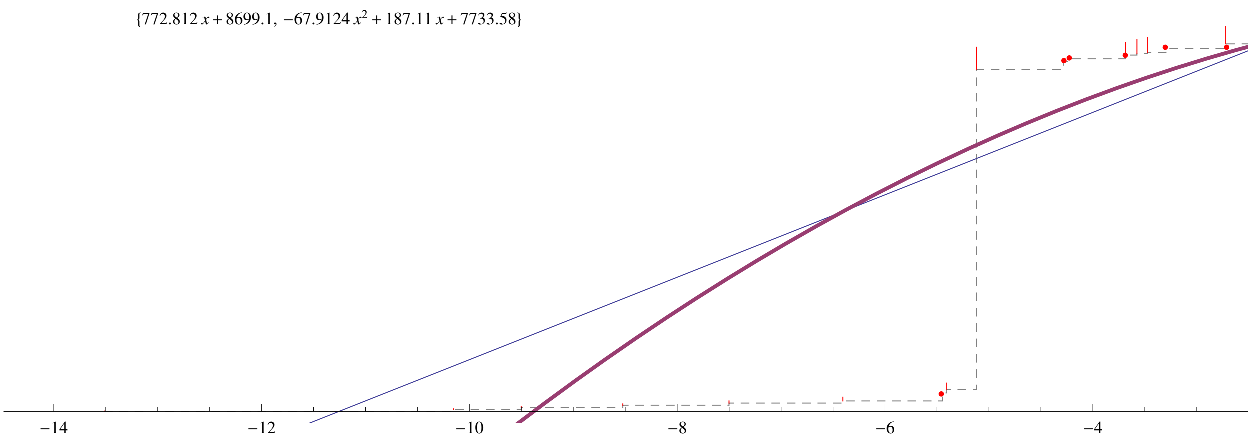
Tough examples

```
examples2 = Sort@{48 511, 39 015, 73, 4888, 328, 1625, 3254, 2045, 1927, 13 708, 35 331};
plotDonations /@ examples2 // TableForm
```

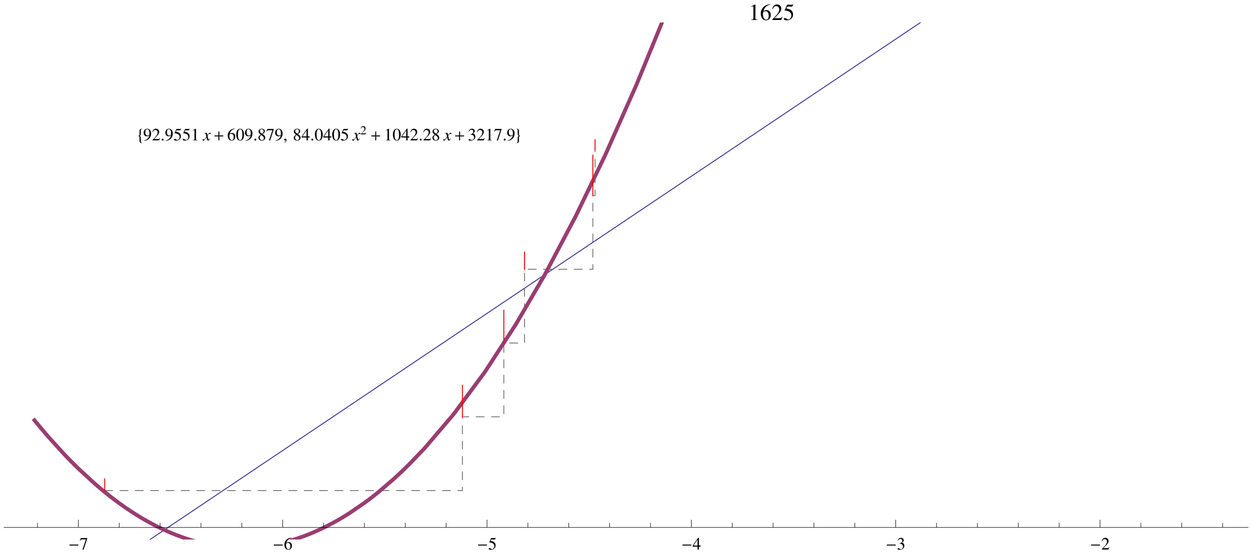
73



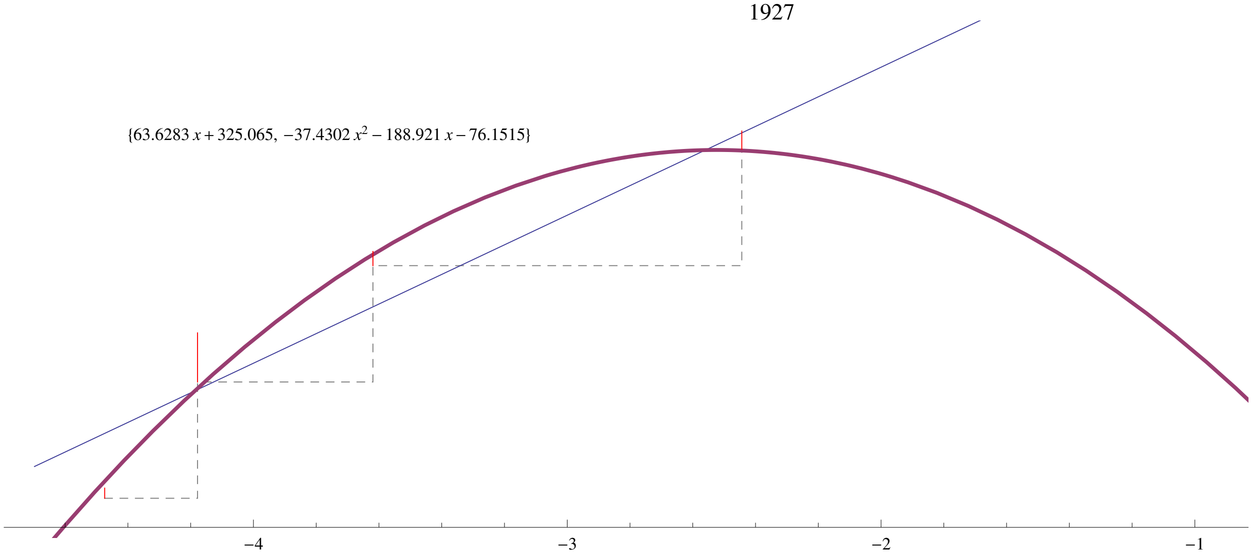
328



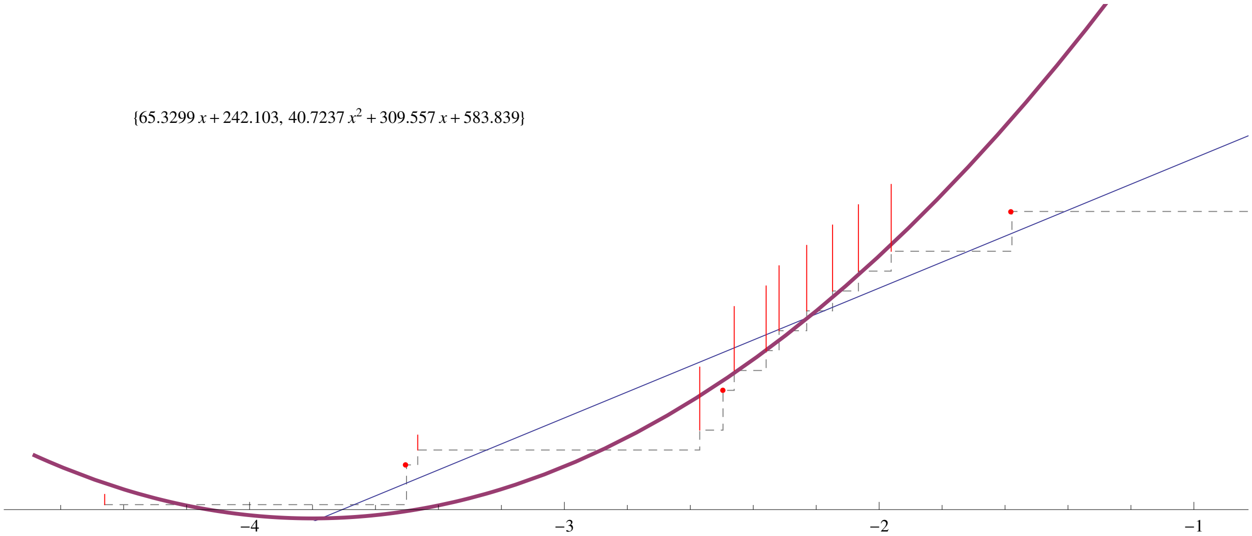
1625



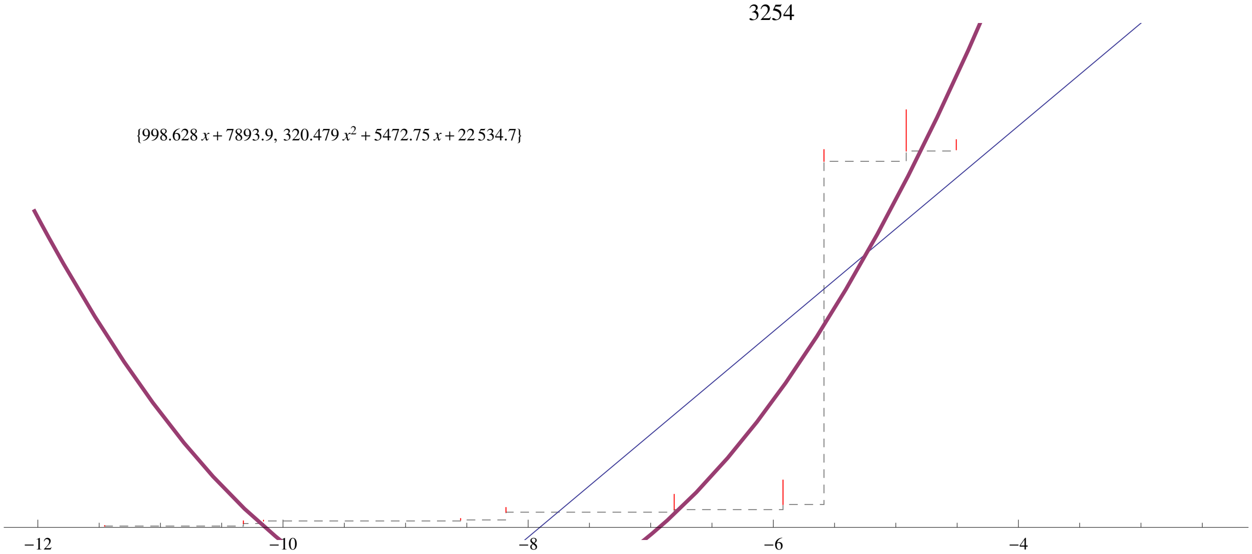
1927



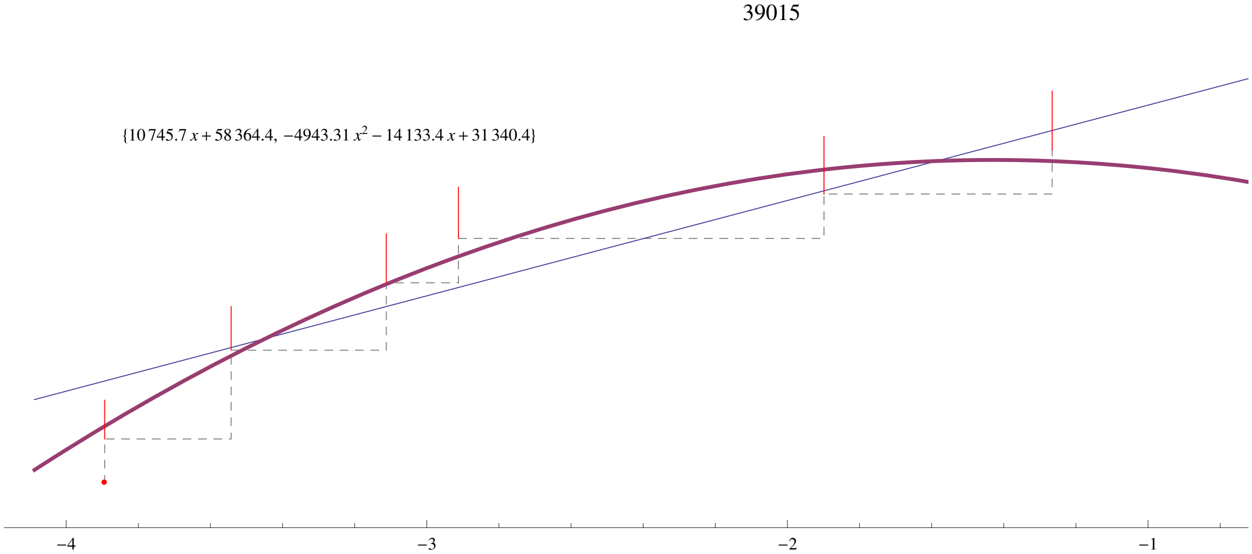
2045

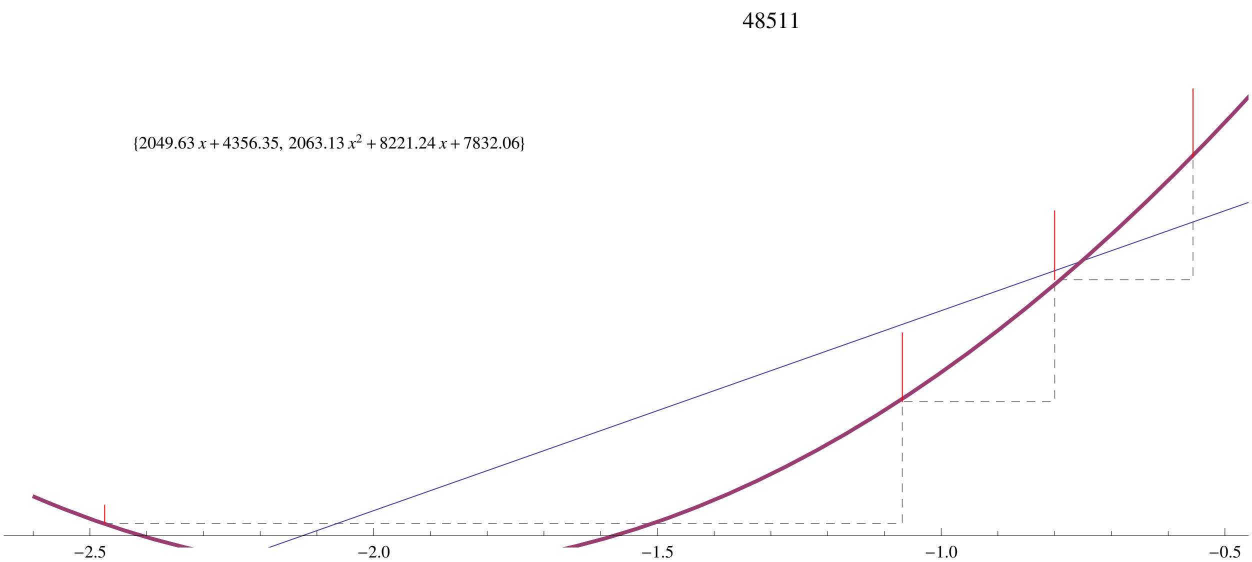


3254



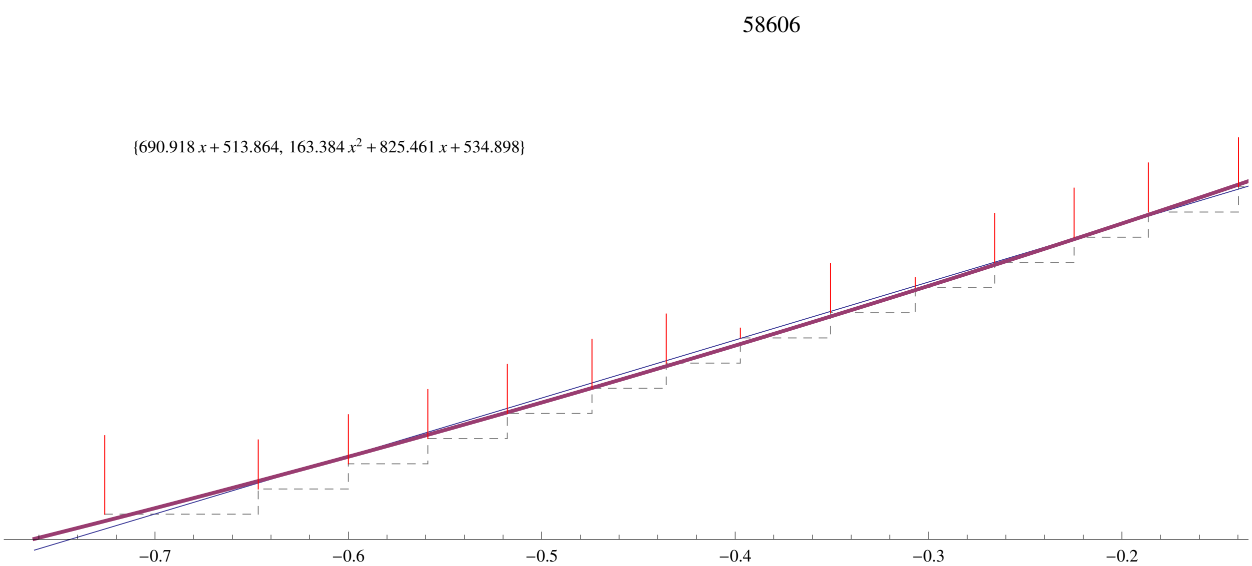
39015





Testing

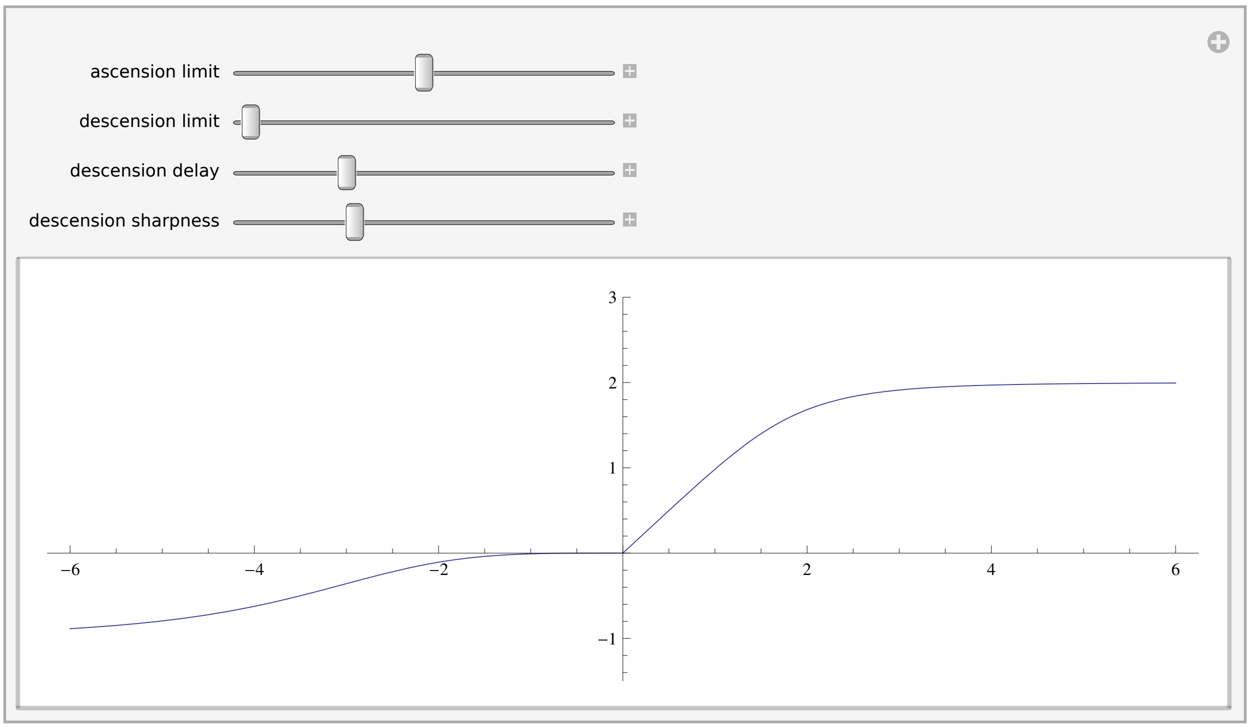
```
plotDonations /@ {58 606} // TableForm
```



Quadratic tempering function

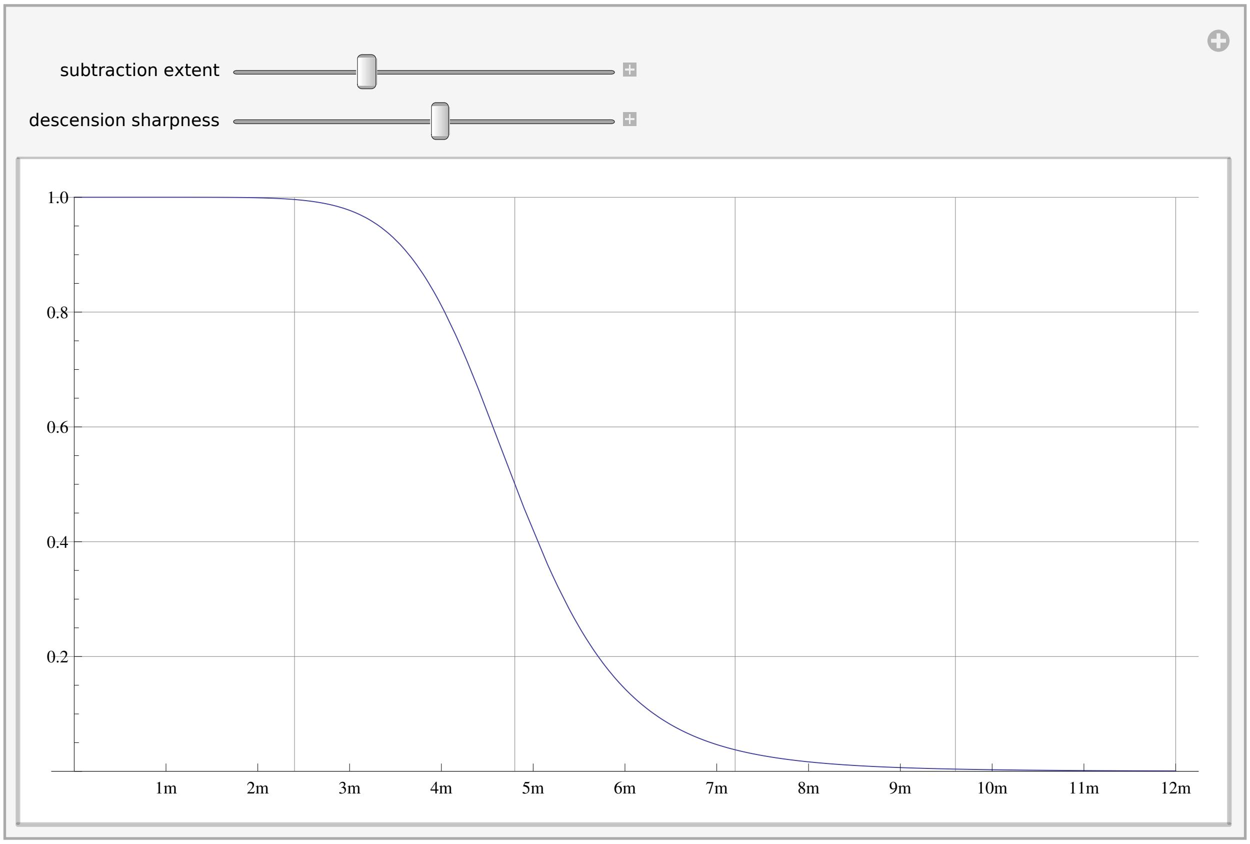
$$\texttt{qtFunc} := \texttt{qt} :\mapsto \begin{cases} \texttt{fqtdlim} - \dfrac{\texttt{fqtdlim}}{1+(-\texttt{i}/\texttt{fqtddel})^{\texttt{fqtdshp}}} & \texttt{i} < 0 \\ \dfrac{\texttt{i}}{\left(1+(\texttt{i}/\texttt{fqtalim})^{4}\right)^{1/4}} & \texttt{i} \geq 0 \end{cases};$$

```
Manipulate[
  Plot[qt, {i, -6, 6}, PlotRange → {-1.5, 3}, AspectRatio → 1 / 3, ImageSize → 600],
  {{fqtalim, 2, "ascension limit"}, 1, 3},
  {{fqtdlim, -1, "descension limit"}, -1, -0.5},
  {{fqtddel, 3.5, "descension delay"}, 1, 10},
  {{fqtdshp, 3.8, "descension sharpness"}, 2, 8}] /. qtFunc
```



Recency subtraction

```
subFunc := sub :→ 1 / (1 + (ybn / fsubext)^fsubshp);
Manipulate[
  Plot[sub, {ybn, 0, 1}, PlotRange → {0, 1},
   AspectRatio → 1 / 2, ImageSize → 600, GridLines → Automatic,
   Ticks → {{# / 12, ToString@# ~~ "m"} & /@ Range[12], Automatic}],
  {{fsubext, 0.5, "subtraction extent"}, 0.1, 1},
  {{fsubshp, 5.8, "descension sharpness"}, 2, 13}] /. subFunc
```



Stretch function

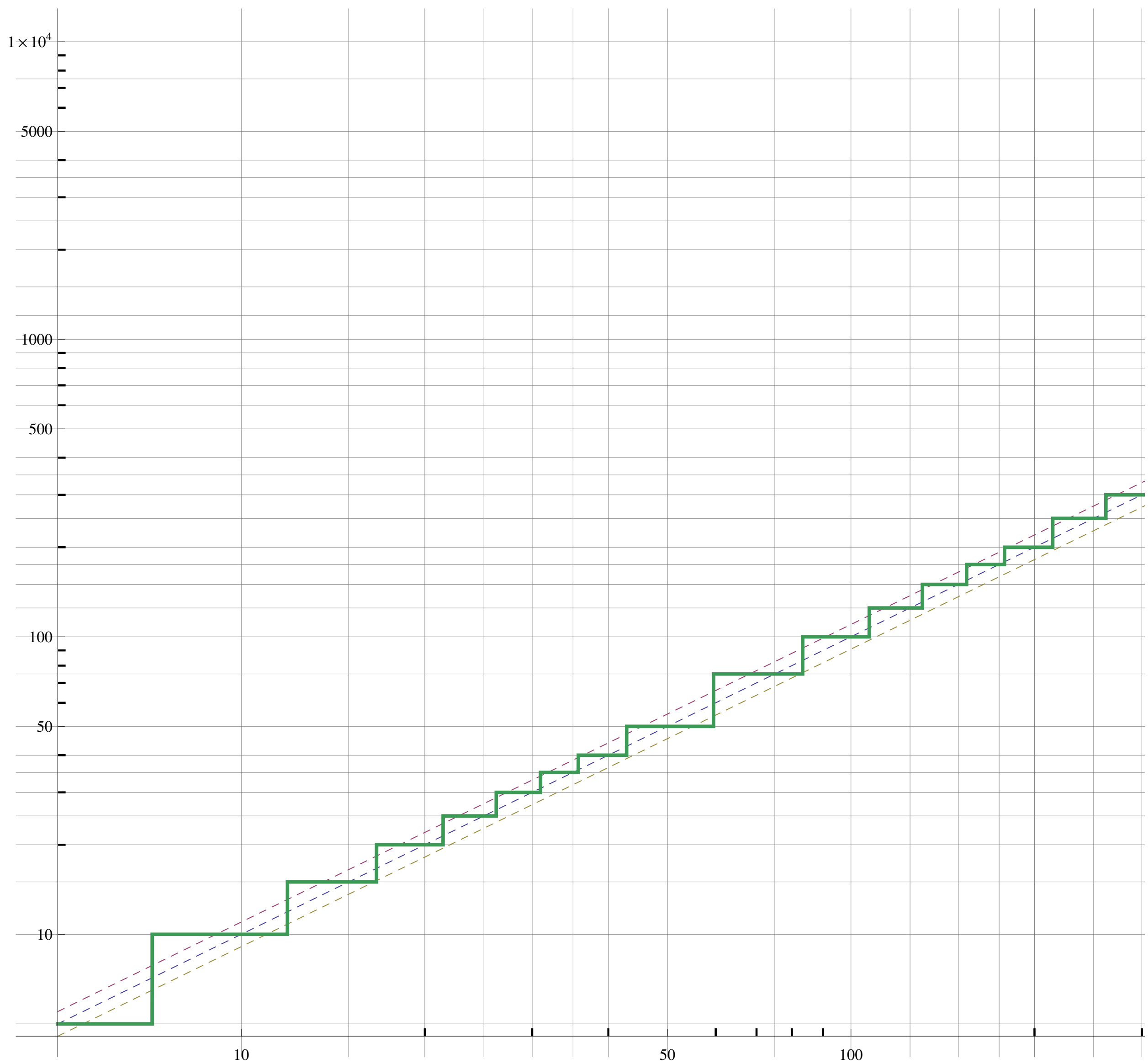
ask amount is stretched by a factor of 1.2

Rounding function

Qualitative description

The rounding function takes a raw ask amount value and converts it to a round ask amount value. For example, if, based on all the complex math, we form a raw ask amount value of $247.11, then we'll ask for a round $250. Similaryly, a raw value of $513.02 might get converted to $500. The raw value might go up or down, depending on where the nearest round value is.

```
roundValues = {5, 10, 15, 20, 25, 30, 35, 40, 50, 75, 100,
   125, 150, 175, 200, 250, 300, 350, 400, 500, 600, 700, 800, 900,
   1000, 1200, 1500, 2000, 2500, 3000, 3500, 4000, 5000, 7500, 10 000};
makeRound = Function[val, First@SortBy[roundValues, Abs[val * 1.05 - #] &]];
LogLogPlot[{x, 1.1 x, x / 1.1, makeRound[x]}, {x, 5, 10 000}, PlotPoints → 500,
 ImageSize → 1200, AspectRatio → 1 / 2, GridLines → {roundValues, roundValues},
 PlotStyle → {{Dashed}, {Dashed}, {Dashed}, {Thick}}]
```



Likelihood algorithm

Qualitative description

The likelihood is a number between 0 and 1, where 0 means very unlikely to give and 1 means very likely to give. It is based on the contributions, specifically the **timing** of the sequence of contributions as well as the **contribution type** of each one.

At the outer-most level of the algorithm, the likelihood is the product of two factors called: **dedication**, and **readiness**.

Dedication: (valued between 0 and 1) If a bundle has been giving frequently and recently, it will have high dedication. Let us consider the affect of a single contribution on the dedication... Immediately after the contribution, the dedication will be high. But as time passes, the dedication will decay slowly and approach zero as time goes to infinity. The **dedication decay function** is responsible for this behavior. But what if a second contribution is received? At the point in time immediately before the second contribution comes, the dedication value (as decayed from the first contribution) is noted. Then, using this pre-contribution dedication value, a post-contribution dedication value is calulated using the **dedication bolster function**. Thus, the dedication has distinct values *before* the contribution and *after*, but at the instant the contribution is made, the dedication is undefined. Immediately after the contribution, the dedication will have risen, and then, from its new position, it will begin a *new* decay process.

This sequence of decay-bolster-decay-bolster continues sequentially across all contributions for the particular bundle until the *current* dedication value is found (using the point in time when the search is run), as decayed from the last contribution. Thus, the calculating the dedication is an iterative process that must be performed in sequence across all contributions. When a bundle does not have any contributions, the dedication value is a function of the time since the first log entry (with the highest dedication being immediately after the first log entry, and decaying afterwards).

Readiness: (valued between 0 and 1) After a donor gives, they're not likely to give again for some time. This logic is the basis for the second likelihood factor, the readiness. Important contributions will force the readiness down to zero for a breif period of time. In turn, this forces the likelihood down to zero (due to multiplication), even though the dedication will be high immediately after the contribution. The readiness will rise to 1 much more quickly than the dedication will decay to zero, though. The readiness for a bundle is the product of all the **specific readiness** values (which also range from 0 to 1), as determined by each contribution. The **specifc readiness function** determines the specific readiness from one contribution.

Constants (fixed values)

These values of these constants are fixed for the duraton of the search. The values are chosen through theoretical speculation, based on interacting with the Manipulate graphs below to produce sensible results.

fdi	initial dedication value
frt	reliability decay time, in years
fdsr	dedication decay sharpness when fully reliable
fdsu	dedication decay sharpness when fully unreliable
fdwr	dedication decay weight when fully reliable
fdwu	dedication decay weight when fully unreliable
fbvz	bolster value at zero -- when optimism (co) is 1 and pre-contribution dedication (cdb) is 0
fbsz	bolster slope at zero -- slope of cda vs cdb when co=1 and cdb=0
fbso	bolster slope at one -- slope of cda vs cdb when co=1 and cdb=1
frdcs	delayed readiness curve start
frdce	delayed readiness curve end

Variables

These variable names will be used consistently in functions throughout the rest of this text. Due to the iterative nature of calculating the likelihood, some variables are specific to a context within the sequence of contributions for a given bundle.

at an arbitrary point in time

ta	years between now and the date of the most recently prior affirmative event
tc	years between now and the date of the most recently prior contribution
rb	reliability

for the current contribution

cybn	years before now
cop	optimism of the current contribution
crs	resilience of the current contribution
cta	years between the time of the current contribution and the most recently prior affirmative event
ctc	years between the time of the current contribution and the most recently prior contribution
cti	years between the time of the current contribution and the first (initial) log entry
crb	reliability at the time of the current contribution
cdbr	dedication immediately before the current contribution, if rb=1
cdbu	dedication immediately before the current contribution, if rb=0
cdb	dedication immediately before the current contribution
cda	dedication immediately after the current contribution
cbs	bolster strength of the current contribution and its point in time
crdi	immediate readiness of the current contribution
crdd	readiness delay of the current contribution
crds	specific readiness of the current contribution

for the previous contribution

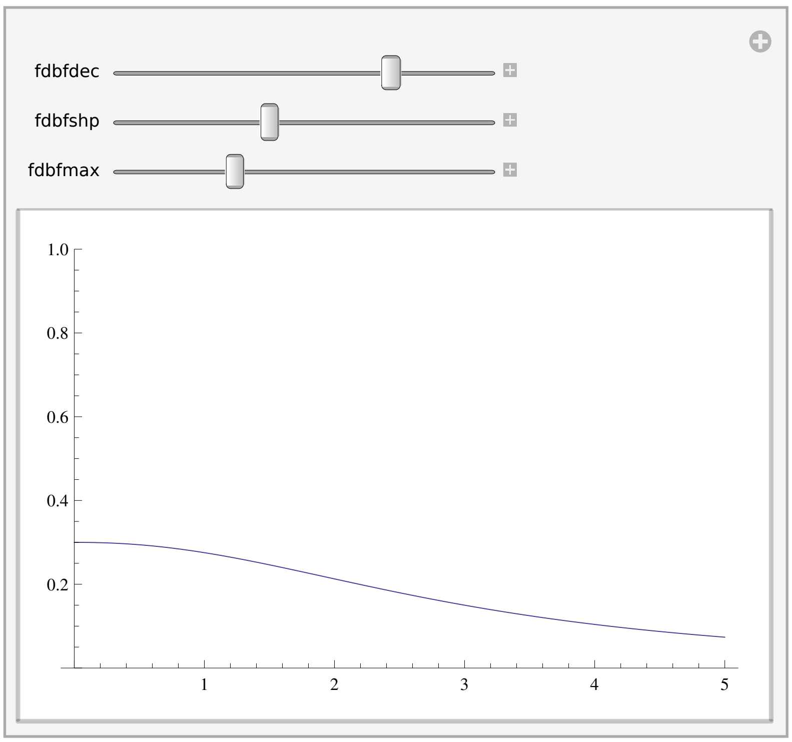
pop	optimism of the previous contribution
prs	resilience of the previous contribution
pta	years between the time of the previous contribution and the most recently prior affirmative event
ptc	years between the time of the previous contribution and the most recently prior contribution
prb	reliability at the time of the previous contribution
pda	dedication immediately after the previous contribution

Functions

Initial dedication value (*cdb* for the first contrib)

$$\texttt{firstCdbFunc := firstCdb :>} \frac{\texttt{fdbfmax}}{1 + (\texttt{cti / fdbfdec})^{\texttt{fdbfshp}}};$$

```
Manipulate[Plot[firstCdb, {cti, 0, 5}, PlotRange → {0, 1}],
  {{fdbfdec, 3}, 0, 4},
  {{fdbfshp, 2.2}, 1, 4},
  {{fdbfmax, 0.3}, 0, 1} ] /. firstCdbFunc
```



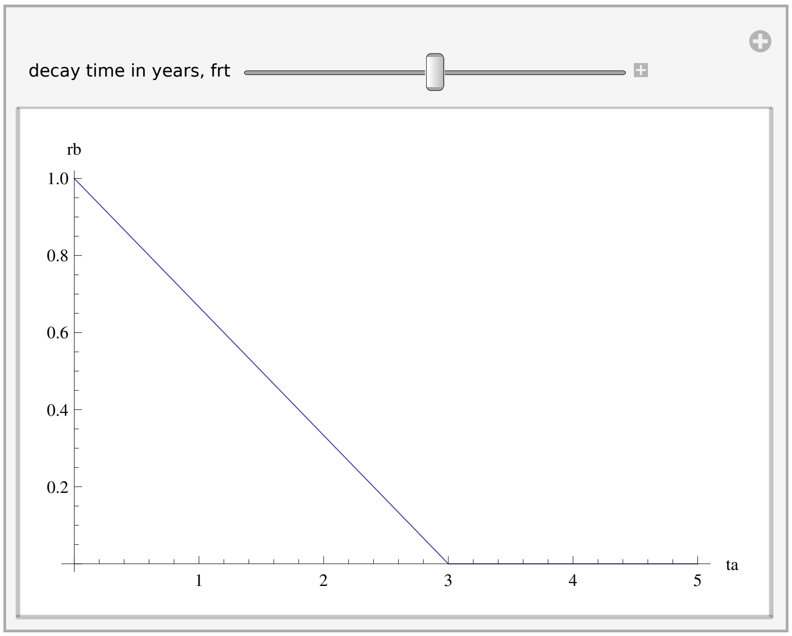
Dedication decay function (to produce *cdb* for subsequent contribs)

Note that this function is contructed to be evaluated from the perspective of the contribution *after* the one that catylizes this dedication decay. Thus, the function does not require any input variables from the “current” contribution -- it just produces output of the variable db (dedication immediately before the current contribution). And this function *does* require input from the variables of the “previous” contribution (the contribution actually responsible for producing the decay curve).

Reliability decay

```
rbFunc := rb :→ If[ta > frt, 0, 1 - ta / frt];
prbFunc := rbFunc /. {rb → prb, ta → pta};

Manipulate[Plot[#, {ta, 0, 5}, AxesLabel → {"ta", "rb"}],
    {{frt, 3, "decay time in years, frt"}, 2, 4}, SaveDefinitions → True] &@rb /.
 rbFunc
```



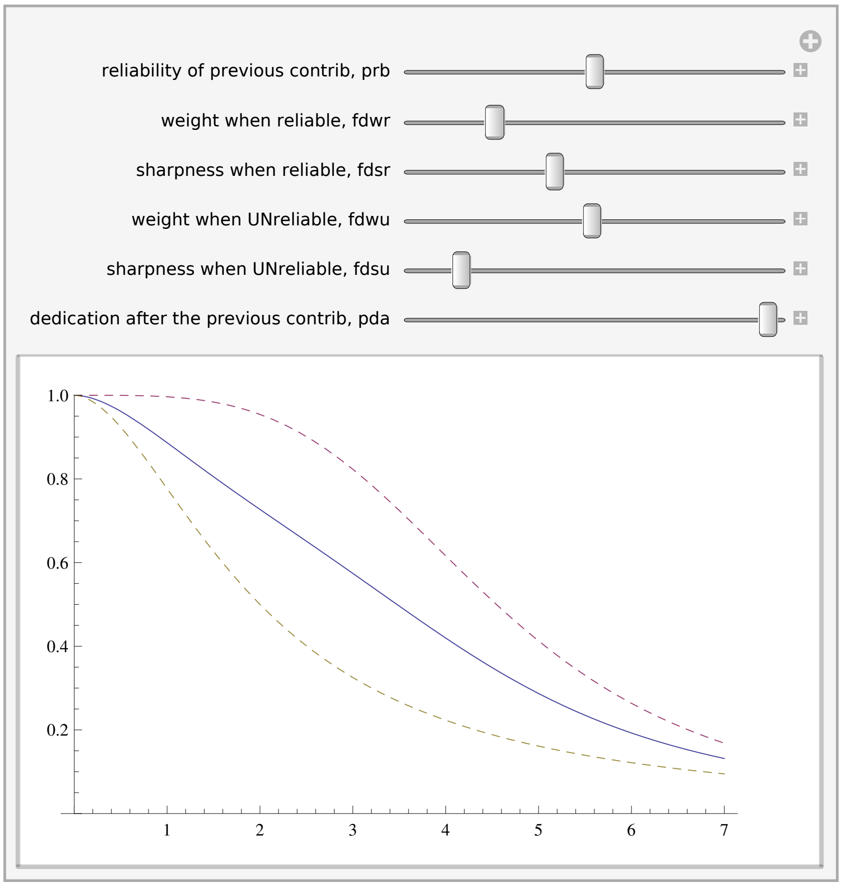
Dedication decay function, with manipulated contants

$$\text{cdbrFunc} := \text{cdbr} :\to \frac{\text{pda}}{1 + (\text{fdwr ctc})^{\text{fdsr}}};$$

$$\text{cdbuFunc} := \text{cdbu} :\to \frac{\text{pda}}{1 + (\text{fdwu ctc})^{\text{fdsu}}};$$

```
cdbFunc := cdb :→ cdbu + (cdbr - cdbu) prb;
```

```
Manipulate[
    Plot[#, {ctc, 0, 7}, PlotRange → {0, 1}, PlotStyle → {Automatic, Dashed, Dashed}],
    {{prb, 0.5, "reliability of previous contrib, prb"}, 0, 1},
    {{fdwr, 0.22, "weight when reliable, fdwr"}, 0.01, 1},
    {{fdsr, 3.7, "sharpness when reliable, fdsr"}, 1, 8},
    {{fdwu, 0.5, "weight when UNreliable, fdwu"}, 0.01, 1},
    {{fdsu, 1.8, "sharpness when UNreliable, fdsu"}, 1, 8},
    {{pda, 1, "dedication after the previous contrib, pda"}, 0, 1},
    SaveDefinitions → True] &@{
  cdb /. cdbFunc /. cdbrFunc /. cdbuFunc,
  cdbr /. cdbrFunc,
  cdbu /. cdbuFunc }
```



Dedication decay function, in various forms

```
cdbFunc
```

cdb :→ cdbu + (cdbr − cdbu) prb

```
cdbFunc /. cdbrFunc /. cdbuFunc
```

$$\text{cdb} :\to \frac{\text{pda}}{1 + (\text{fdwu ctc})^{\text{fdsu}}} + \left(\frac{\text{pda}}{1 + (\text{fdwr ctc})^{\text{fdsr}}} - \frac{\text{pda}}{1 + (\text{fdwu ctc})^{\text{fdsu}}}\right) \text{prb}$$

```
Expand[(cdb /. cdbFunc /. cdbrFunc /. cdbuFunc) / pda] * pda
```

$$\mathrm{pda}\left(\frac{1}{1+(\mathrm{ctc}\ \mathrm{fdwu})^{\mathrm{fdsu}}}+\frac{\mathrm{prb}}{1+(\mathrm{ctc}\ \mathrm{fdwr})^{\mathrm{fdsr}}}-\frac{\mathrm{prb}}{1+(\mathrm{ctc}\ \mathrm{fdwu})^{\mathrm{fdsu}}}\right)$$

```
cdbFunc /. cdbrFunc /. cdbuFunc /. prbFunc
```

$$\mathrm{cdb} :\to \frac{\mathrm{pda}}{1+(\mathrm{fdwu}\ \mathrm{ctc})^{\mathrm{fdsu}}}+\left(\frac{\mathrm{pda}}{1+(\mathrm{fdwr}\ \mathrm{ctc})^{\mathrm{fdsr}}}-\frac{\mathrm{pda}}{1+(\mathrm{fdwu}\ \mathrm{ctc})^{\mathrm{fdsu}}}\right)\mathrm{If}\left[\mathrm{pta}>\mathrm{frt},\ 0,\ 1-\frac{\mathrm{pta}}{\mathrm{frt}}\right]$$

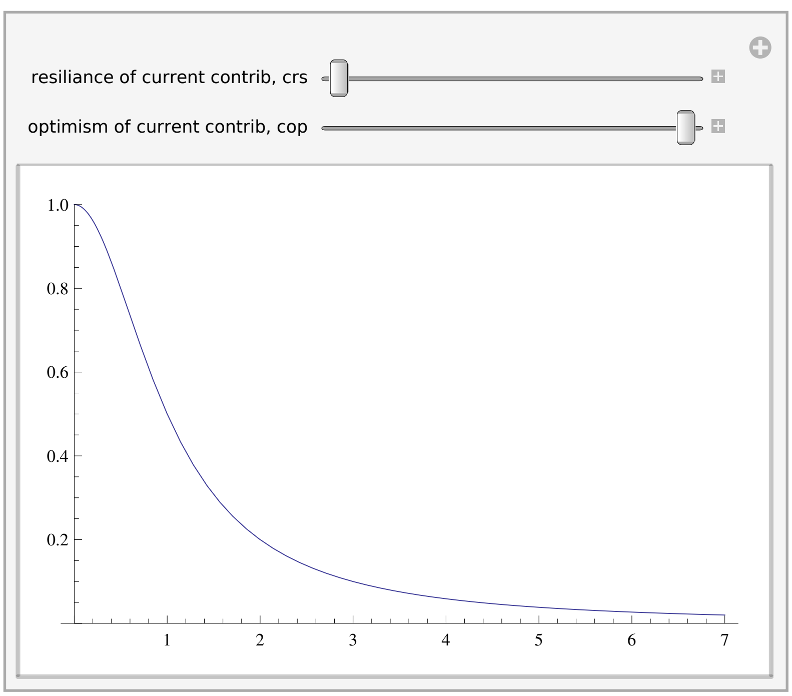
Dedication bolster function (to produce the *cda* value

Bolster strength function

Resilient contributions maintain their strength even long after the last affirmative event. Contributions with zero resilience, will have only about 1/2 the

$$\mathrm{cbsFunc} := \mathrm{cbs} :\to \frac{\mathrm{cop}}{1+(1-\mathrm{crs})^{4}\,\mathrm{cta}^{2}};$$

```
Manipulate[Plot[#, {cta, 0, 7}, PlotRange → {0, 1}],
    {{crs, 0, "resiliance of current contrib, crs"}, 0, 1},
    {{cop, 1, "optimism of current contrib, cop"}, 0, 1},
    SaveDefinitions → True] &@cbs /. cbsFunc
```



Dedication bolster function form

```
f[x_] := a x^3 + b x^2 + c x + d;
```

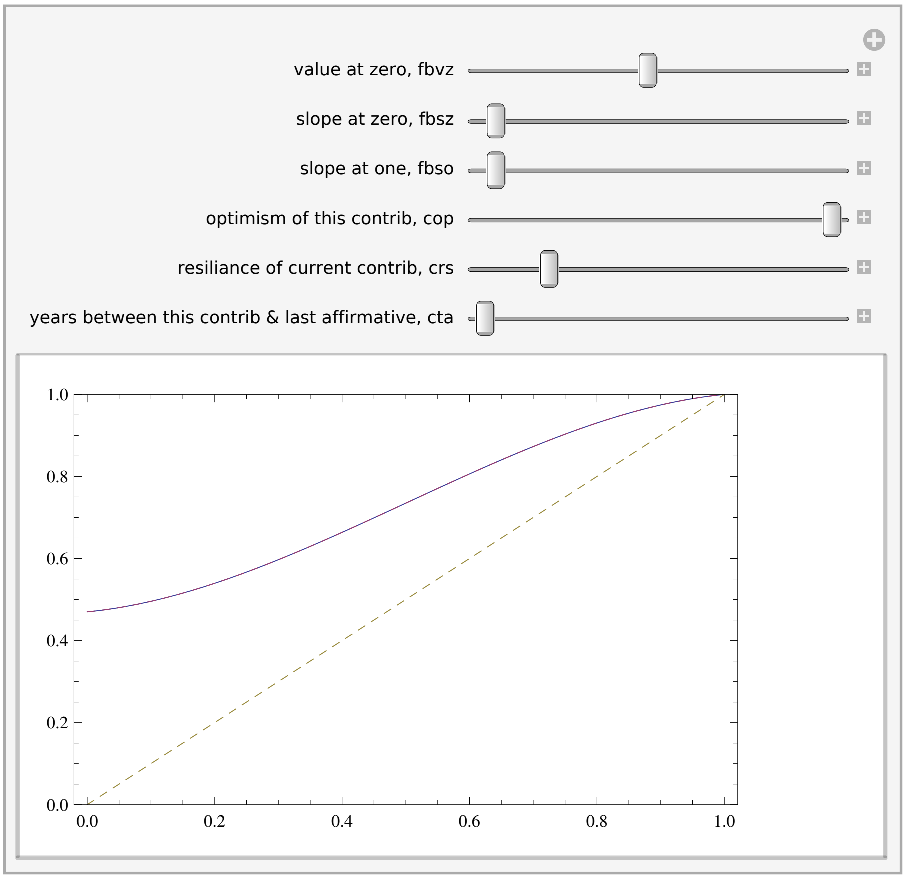
```
optimisticBolster = f[cdb] /.
  Solve[f[0] == fbvz && f[1] == 1 && f'[0] == fbsz && f'[1] == fbso, {a, b, c, d}][[1]]
```

$\mathrm{cdb}\ \mathrm{fbsz} + \mathrm{cdb}^2\ (3 - \mathrm{fbso} - 2\ \mathrm{fbsz} - 3\ \mathrm{fbvz}) + \mathrm{fbvz} + \mathrm{cdb}^3\ (-2 + \mathrm{fbso} + \mathrm{fbsz} + 2\ \mathrm{fbvz})$

Dedication bolster function, with manipulated contants

```
cdaFunc := cda → cdb + (optimisticBolster - cdb) cbs;
```

```
Manipulate[Plot[#, {cdb, 0, 1}, PlotRange → {0, 1},
     Frame → True, PlotStyle → {Automatic, Dashed, Dashed}],
    {{fbvz, 0.47, "value at zero, fbvz"}, 0, 1},
    {{fbsz, 0.15, "slope at zero, fbsz"}, 0, 5},
    {{fbso, 0.15, "slope at one, fbso"}, 0, 5},
    {{cop, 0.5, "optimism of this contrib, cop"}, 0, 1},
    {{crs, 0, "resiliance of current contrib, crs"}, 0, 1},
    {{cta, 0, "years between this contrib & last affirmative, cta"}, 0, 7},
    SaveDefinitions → True] &@{
   cda /. cdaFunc /. cbsFunc,
   optimisticBolster,
   cdb}
```



Dedication bolster function, in various forms

```
cdaFunc // FullSimplify
```

cda →
 cdb + cbs (fbvz + cdb ((-1 + cdb) (1 - fbsz + cdb (-2 + fbso + fbsz)) + cdb (-3 + 2 cdb) fbvz))

```
cda /. cdaFunc // Expand
```

$cdb - cbs\ cdb + 3\ cbs\ cdb^2 - 2\ cbs\ cdb^3 - cbs\ cdb^2\ fbso + cbs\ cdb^3\ fbso + cbs\ cdb\ fbsz - 2\ cbs\ cdb^2\ fbsz + cbs\ cdb^3\ fbsz + cbs\ fbvz - 3\ cbs\ cdb^2\ fbvz + 2\ cbs\ cdb^3\ fbvz$

```
Variables[cda /. cdaFunc]
```

{cdb, cbs, fbsz, fbso, fbvz}

```
cdaFunc /. cbsFunc // Simplify
```

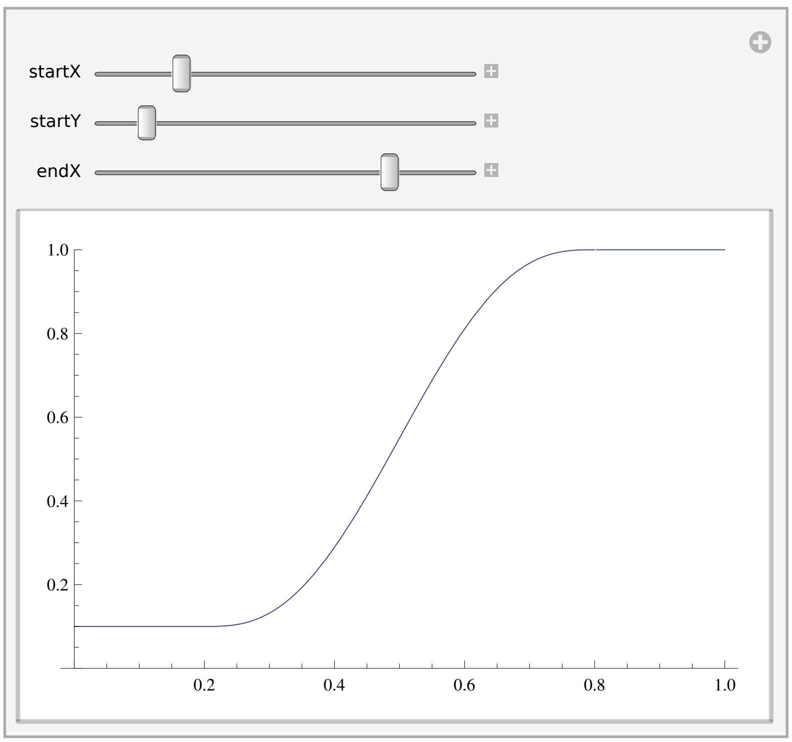
$$\text{cda} \to \text{cdb} + \left((-1 + \text{cdb})\ \text{cop}\left(-\text{fbvz} - \text{cdb}\ (-1 + \text{fbsz} + \text{fbvz}) + \text{cdb}^2\ (-2 + \text{fbso} + \text{fbsz} + 2\ \text{fbvz})\right)\right) \Big/ \left(1 + (-1 + \text{crs})^4\ \text{cta}^2\right)$$

```
Variables[cda /. cdaFunc /. cdbFunc /. cdbrFunc /. cdbuFunc /. cbsFunc]
```

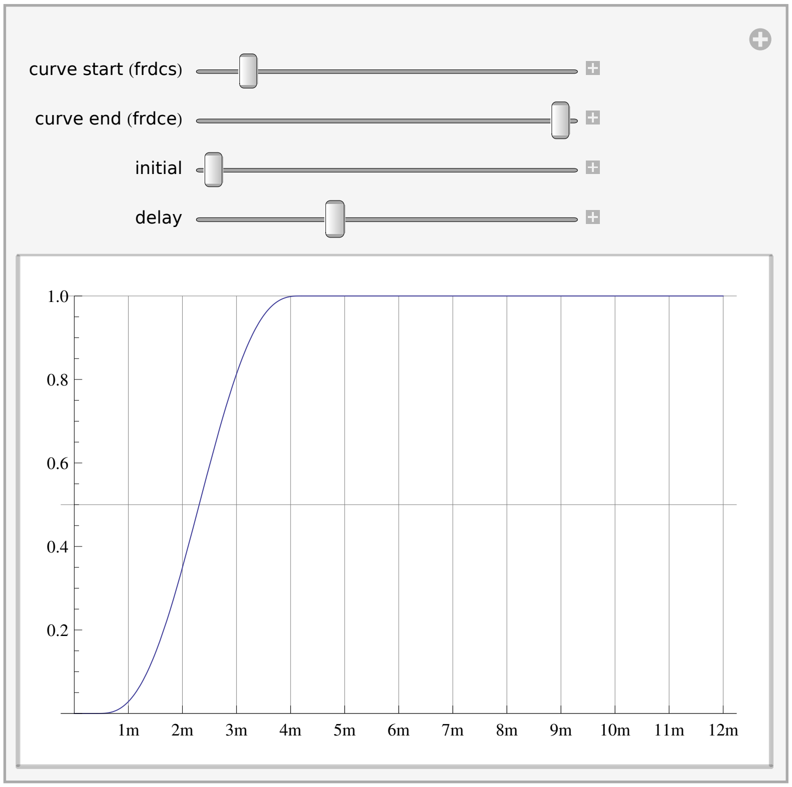
$\left\{\text{cop}, \text{crs}, \text{cta}, \text{fbso}, \text{fbsz}, \text{fbvz}, (\text{ctc fdwr})^{\text{fdsr}}, (\text{ctc fdwu})^{\text{fdsu}}, \text{pda}, \text{prb}\right\}$

Readiness function (to produce the *crds* value)

```
f = c0 + c1 # + c2 #^2 + c3 #^3 + c4 #^4 + c5 #^5 &;
sol = Solve[f[0] == 0 && f'[0] == 0 && f''[0] == 0 && f[1] == 1 && f'[1] == 0 && f''[1] == 0,
    {c0, c1, c2, c3, c4, c5}];
f2 = f /. sol[[1]];
f3 = (1 - startY) * f2[1/(endX - startX) (# - startX)] + startY &;
f4 = If[# < startX, startY, If[# > endX, 1, f3[#]]] &;
f4 = Piecewise[{{startY, # < startX}, {1, # > endX}}, f3[#]] &;
Manipulate[Plot[#, {x, 0, 1}, PlotRange → {0, 1}],
    {{startX, 0.2}, 0, 1},
    {{startY, 0.1}, 0, 1},
    {{endX, 0.8}, 0, 1},
    SaveDefinitions → True] &@f4[x]
```



```
crds = f4[#] /. {startY → crdi, startX → crdd * frdcs, endX → crdd * frdce} &;
Manipulate[Plot[#, {ybn, 0, 1},
     PlotRange → {0, 1}, GridLines → {Range[0, 1, 1 / 12], {0, 1 / 2, 1}},
     Ticks → {{# / 12, ToString@# ~~ "m"} & /@ Range[12], Automatic}],
    {{frdcs, 0.1, "curve start (frdcs)"}, 0, 1},
    {{frdce, 1, "curve end (frdce)"}, 0, 1},
    {{crdi, 0, "initial"}, 0, 1},
    {{crdd, 1, "delay"}, 0, 1},
    SaveDefinitions → True] &@crds[ybn]
```




```
crds[ybn] // Simplify
```

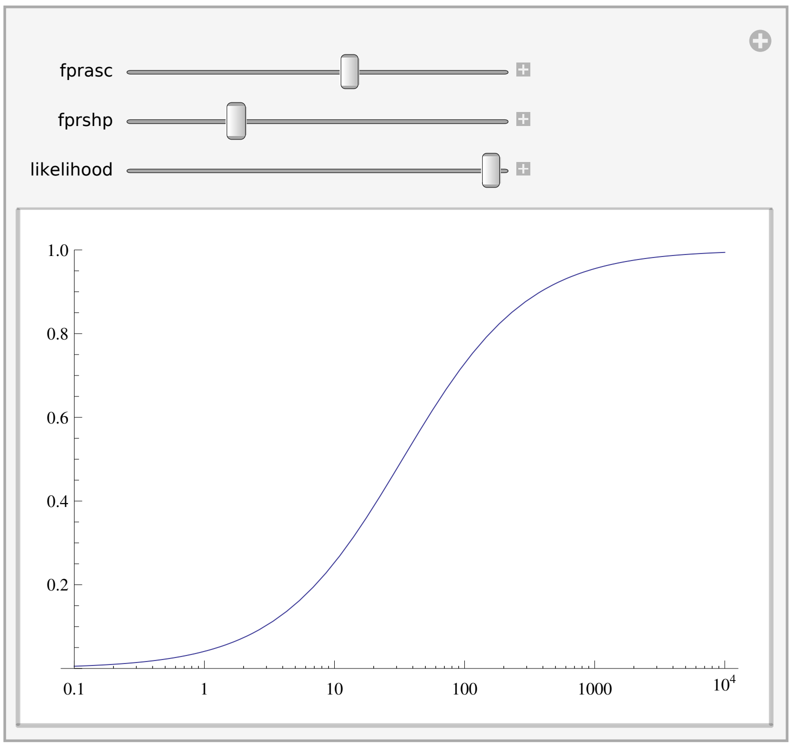
$$\begin{cases} crdi & ybn < crdd\ frdcs \\ 1 & ybn > crdd\ frdce \\ crdi + ((-1 + crdi)\ (crdd\ frdcs - ybn)^3 \ (crdd^2\ (10\ frdce^2 - 5\ frdce\ frdcs + frdcs^2) + 3\ crdd\ (-5\ frdce + frdcs)\ ybn + 6\ ybn^2))\ /\ (crdd^5\ (frdce - frdcs)^5) & True \end{cases}$$

Priority algorithm

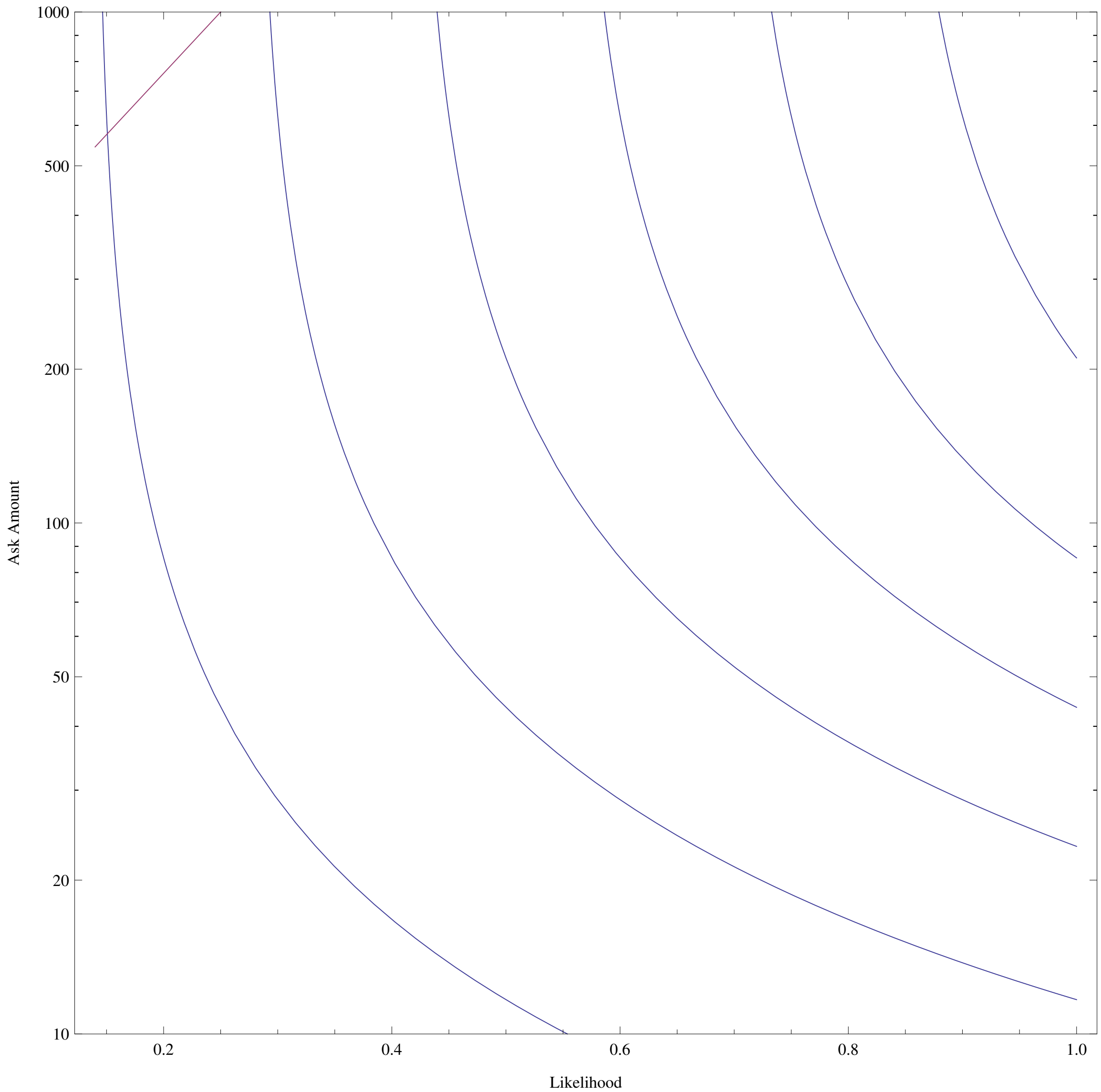
Functions

$$\text{priorityFunc} := \text{priority} :\!\to \text{likelihood}\left(1 - \frac{1}{1 + (\text{fprasc} * \text{aaRaw})^{\text{fprshp}}}\right);$$

```
Manipulate[
  LogLinearPlot[priority, {aaRaw, 0.1, 10 000}, PlotRange → {0, 1}],
  {{fprasc, 0.03}, 0.001, 0.05},
  {{fprshp, 0.9}, 0.5, 2},
  {{likelihood, 1}, 0, 1}
 ] /. priorityFunc
```



```
eqn = (priority /. priorityFunc /. {fprasc → 3 / 100, fprshp → 9 / 10} /.
      {aaRaw → a, likelihood → k}) == p;
exp = a /. Solve[eqn, a][[1]];
LogPlot[
 {Table[exp, {p, 0.14, 1, 0.14}],
  Table[10^(2.4 (k+d)), {d, Range[9]}]
 },
 {k, 0.14, 1}, PlotRange → {10, 1000},
 Frame → True, FrameLabel → {"Likelihood", "Ask Amount"},
 ImageSize → 600, AspectRatio → 1]
```




```
Table[10^(2 (k+d)), {d, {}}]
```

$\{10^{2\ (-0.7+k)},\ 10^{2\ (-0.4+k)},\ 10^{2\ (-0.1+k)},\ 10^{2\ (0.2+k)},\ 10^{2\ (0.5+k)},\ 10^{2\ (0.8+k)},\ 10^{2\ (1.1+k)}\}$

Analysis

Behavior for bundles without contributions

Only relevant for direct mail, so all constants are DM values

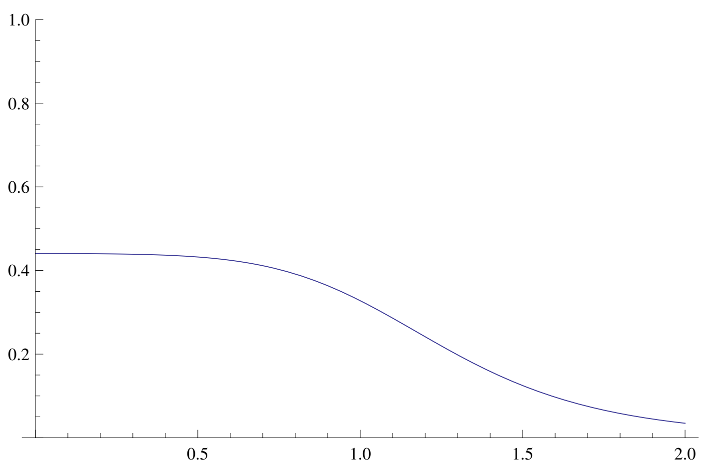
```
baselineFunc
firstCdbFunc
priorityFunc
```

$$\text{baseline} :\to \frac{\text{fblmax}}{1 + \left(\frac{\text{yaf1}}{\text{fbldec}}\right)^{\text{fblshp}}}$$

$$\text{firstCdb} :\to \frac{\text{fdbfmax}}{1 + \left(\frac{\text{cti}}{\text{fdbfdec}}\right)^{\text{fdbfshp}}}$$

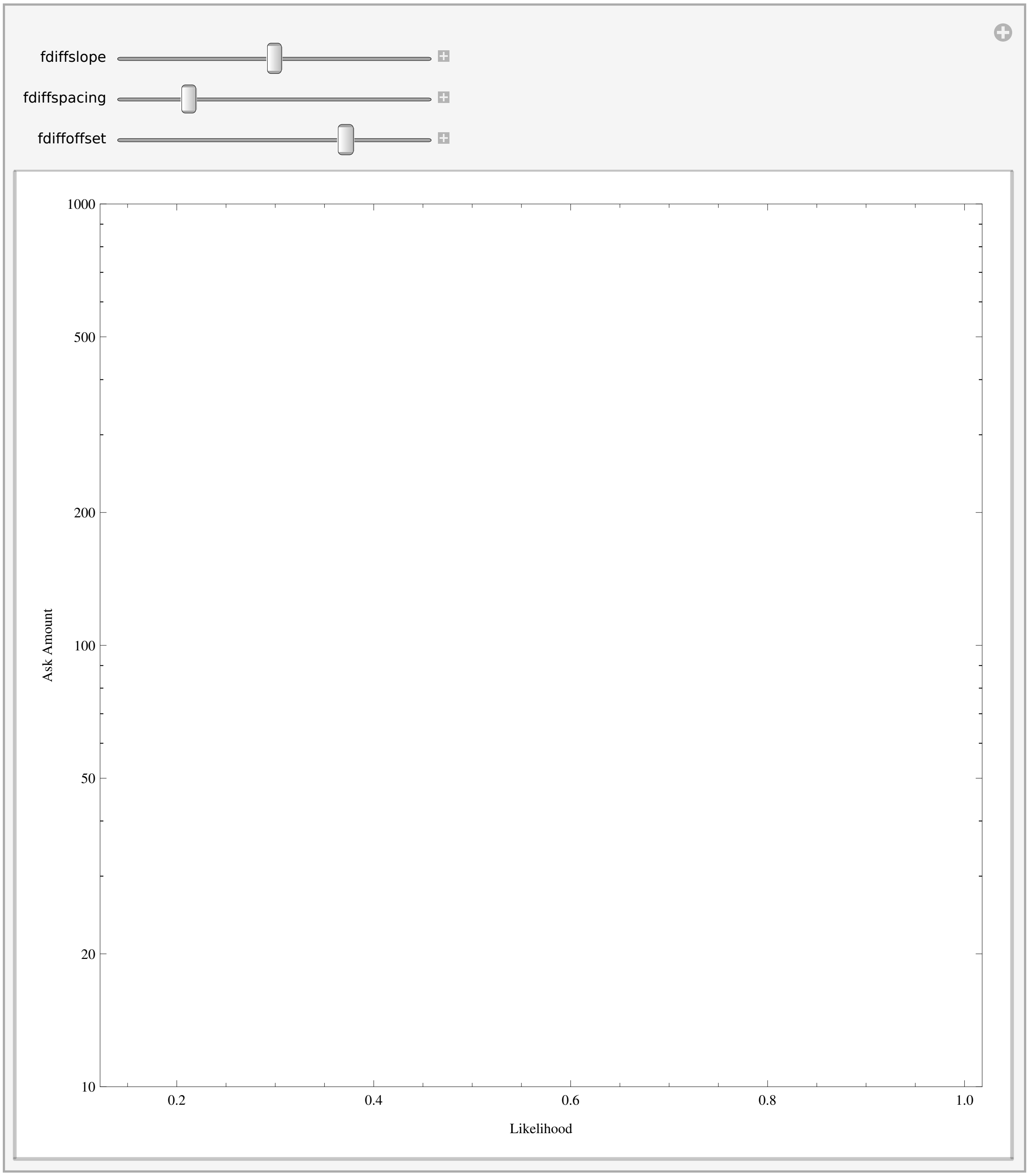
$$\text{priority} :\to \text{likelihood}\left(1 - \frac{1}{1 + (\text{fprasc}\ \text{aaRaw})^{\text{fprshp}}}\right)$$

```
constants = {
   askStretch → 1.2,
   (* baseline ask amount constants *)
   fbldec → 2,
   fblshp → 2.5,
   fblmax → 50,
   (* dedication *)
   fdbfdec → 1.3,
   fdbfshp → 5,
   fdbfmax → 0.7,
   (* priority contstants *)
   fprasc → 0.03,
   fprshp → 0.9
  };
noContribPriority =
  priority /. priorityFunc /. {aaRaw → baseline * askStretch, likelihood -> firstCdb} /.
      baselineFunc /. firstCdbFunc /. {cti → yafl} /. constants;
Plot[noContribPriority, {yafl, 0, 2}, PlotRange → {0, 1}]
```



Difficulty algorithm

```
eqn = (priority /. priorityFunc /. {fprasc → 3 / 100, fprshp → 9 / 10} /.
      {aaRaw → a, likelihood → k}) == p;
exp = a /. Solve[eqn, a][[1]];
Manipulate[LogPlot[
   {Table[exp, {p, 0.14, 1, 0.14}],
    Table[E^(fdiffslope (k+fdiffspacing*d+fdiffoffset)), {d, Range[9]}]
   },
   {k, 0.14, 1}, PlotRange → {10, 1000},
   Frame → True, FrameLabel → {"Likelihood", "Ask Amount"},
   ImageSize → 600, AspectRatio → 1],
 {{fdiffslope, 6}, 5, 7},
 {{fdiffspacing, 0.12}, 0.1, 0.2},
 {{fdiffoffset, -0.4}, -0.7, -0.3}]
```




```
E^(fdiffslope (k+fdiffspacing*d+fdiffoffset)) /.
 {fdiffslope → 6, fdiffspacing → 12 / 100, fdiffoffset → -4 / 10}
```

$e^{6\left(-\frac{2}{5}+\frac{3\,d}{25}+k\right)}$

```
Solve[E^(fdiffslope (k+fdiffspacing*d+fdiffoffset)) == a, d, Reals]
```

```
{{d → ConditionalExpression[(-fdiffoffset fdiffslope - fdiffslope k + Log[a]) /
      (fdiffslope fdiffspacing), a > 0]}}
```

```
dificultyFunc =
 d :> (-fdiffoffset fdiffslope - fdiffslope k + Log[a]) / (fdiffslope fdiffspacing)
```

$$d :\to \frac{-\text{fdiffoffset fdiffslope} - \text{fdiffslope } k + \text{Log}[a]}{\text{fdiffslope fdiffspacing}}$$

Analysis of results

Scrap