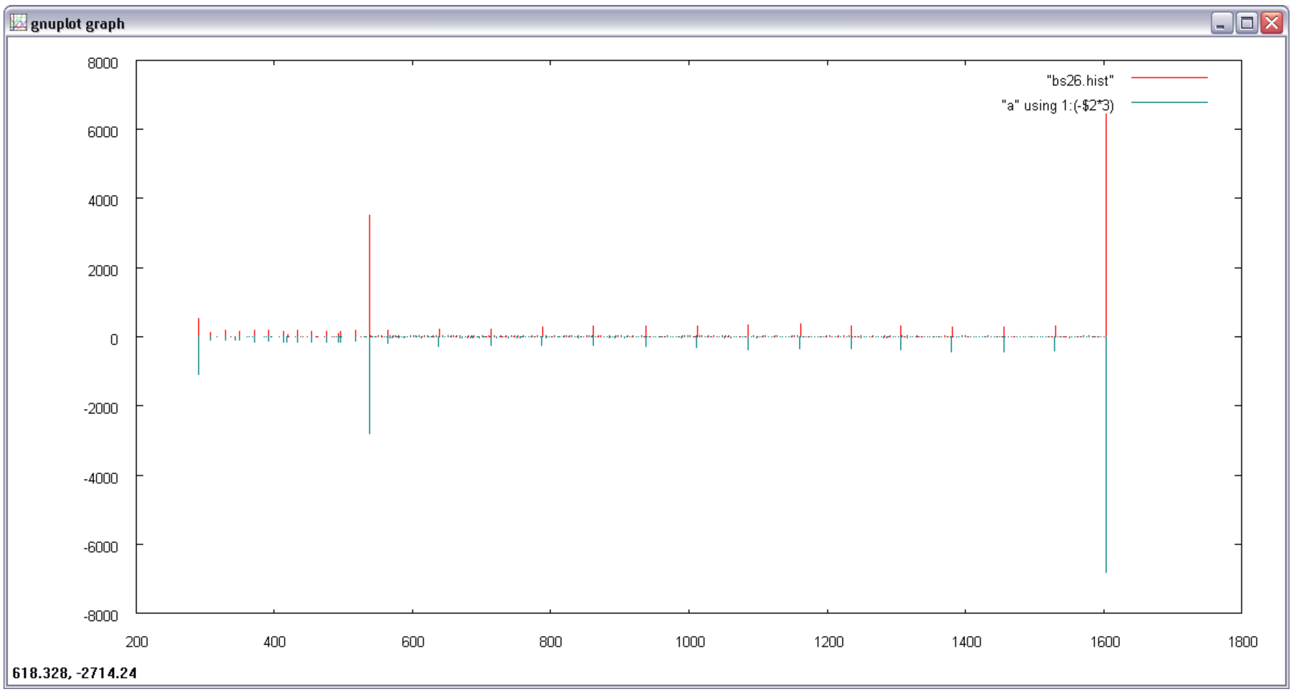
Test case: using a 26 damage dagger (soulskive) with no elmental component so all hits with the same base. Lvl 85 rogue with essentially all AA. Gear was chosen to have no heroic strength and no high ranks of ferocity (although some rogue AA adjust the minimum hits too – precise strikes IIRC).

Parse covered 34905 non-crit backstabs and 5618 crits; parsed over 16 hours.

The basic parse looks like this:



Here the +ve lines from zero upwards are the actual histogram of hit frequencies observed. The -ve lines are as predicted by my model. The model is, as can be seen, not entirely accurate.

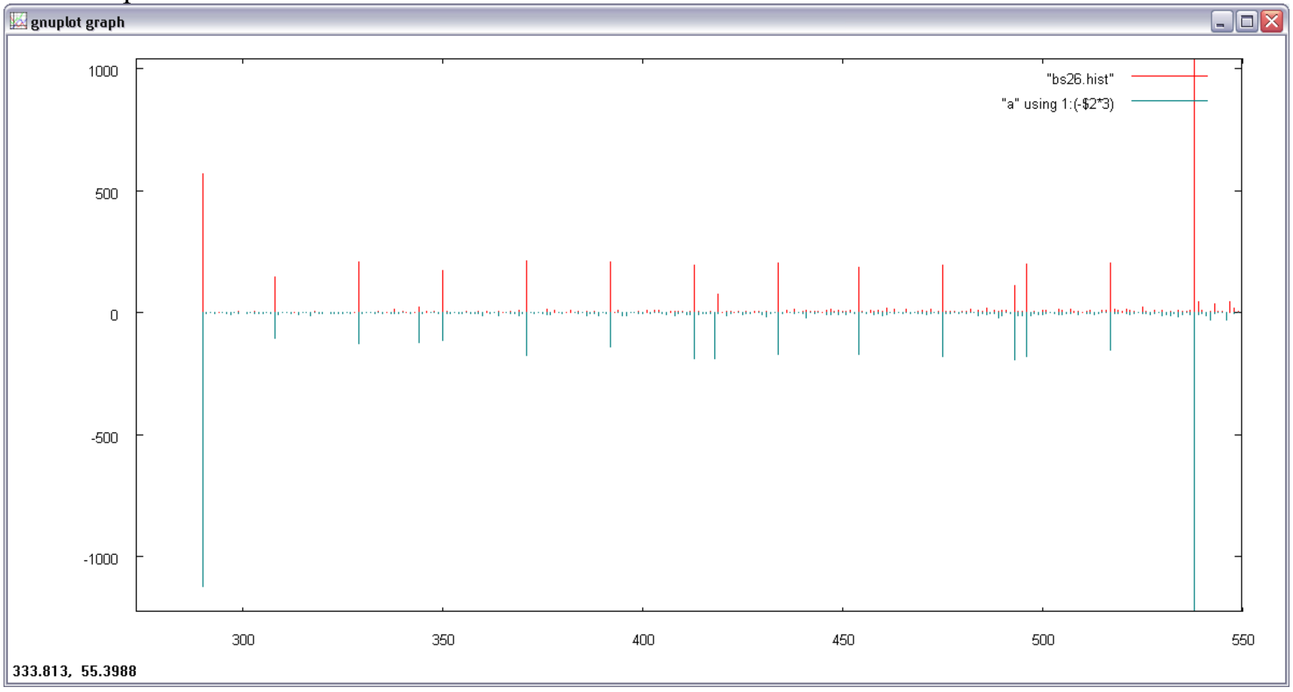
The formula comes from DB + ATK_ROLL[1 to 20]*WD/10*PC_DAM_BONUS as discussed in http://www.thesteelwarrior.org/forum/showthread.php?t=17773.

The PC_DAM_BONUS is random value between 1.00 and 3.55 (at lvl 85) in I belive 0.01 increments. The ATK_ROLL is a whole number between 1 and 20. WD is weapon damage; 26 in this example.

The distribution is dominated by three main spikes, with smaller spikes visible within it. PC_DAM_BONUS is more likely to be at either minimum or maximum value, giving rise to these spikes. Essentially we have three distributions summed together.

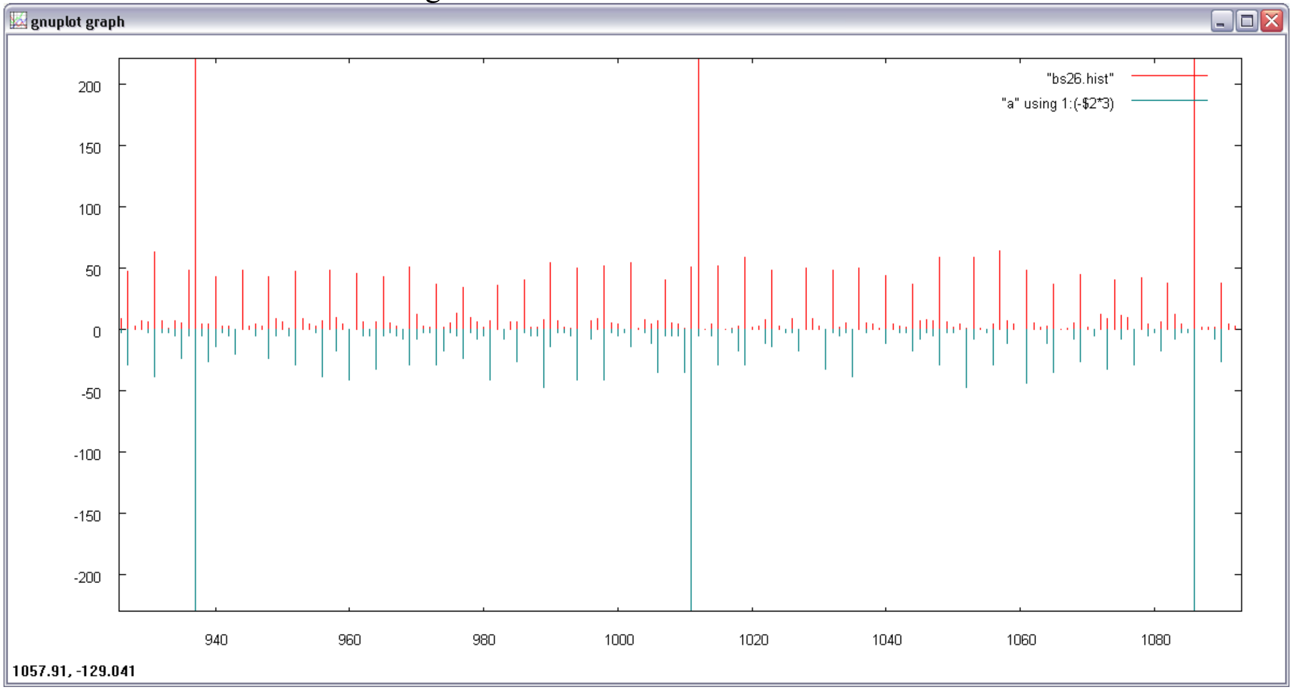
1) PC_DAM_BONUS=1, with ATK_ROLL from 1 to 20. This covers the left and central spikes. The distribution extends from <=290 (clipped) to 538 in this plot.
2) PC_DAM_BONUS=3.55, with ATK_ROLL from 1 to 20. This is the cause of the high spike to the right (ATK_ROLL=20) and the other 19 smaller spikes inbetween the left and right edges.
3) 1 < PC_DAM_BONUS < 3.55. These cause a whole background of hit values. However because ATK_ROLL=20 is more common than the others (as can be seen by the right hand spike being the highest) we therefore also have 255 values in this background that are higher than the rest.

A close up on the left shows:



The left spike here is due to multiple hits stacking up together – 290 is our minimum possible backstab due to level and AA. The right one is what's sometimes referred to as the "2*damage" spike, as that's the roll20*dam/10 spike, with pcdamage=1.

A close on in the middle of the right hand section shows:



We can clearly see 3 of the 20 main spikes corresponding to PC_DAMAGE=max and ATK_ROLLs of 11, 12 and 13.

Underneath this we see the background with varying PC_DAMAGE, but due to ATK_ROLL=20 being more common it also shows spikes. My model evidently has rounding issues still, but the average frequency is about right, indicating the granularity of PC_DAMAGE is 0.01.

The code for my model (and hence formula) in perl is:

```
#!/c/cygwin/bin/perl -w

use strict;

# lvl 80 6.7/2
# lvl 85 7.1/2
my $pcdam_max = 3.55;

# 108 from precise strikes?, 11 from where?
my $db = 120;

# From skill 302 + (str > 200)? Working on more data.
my $bs_mult = 8.03; # between and 8.0195 and 8.0575

#my $db = 20;
#my $bs_mult = 1; # Normal piercing hits

# Backstab damage on weapon
my $wd = 26;

my %hist;
for (my $i = 0; $i < 10000; $i++) {
    # Slightly larger than 1-20 to give a positive skew
    my $roll20 = int(25*rand())+1;

    # Also randomly bias against low values, so we get decay at low end.
    # 99% sure this isn't how it works, but just as a very rough bias.
    while (rand()>0.1) {
      $roll20++;
    }

    # Cap at 1-20 again
    $roll20 =  1 if ($roll20 < 1);
    $roll20 = 20 if ($roll20 > 20);

    # PC damage. Again it should be 1..3.55, but use larger size and clip
    # in order to obtain a skew more akin to what I see. Again this isn't
    # likely a realistic model for how EQ really works.
    my $pcdam = 6*rand();
    $pcdam = 1.0 if ($pcdam < 1);
    $pcdam = $pcdam_max if ($pcdam > $pcdam_max);

    # And now the actual hit formula.
    # NB: There's a lot of rounding going on here, but it's still not quite
    # correct.
    # The int(100*pcdam)/100+0.5 component is just rounding the pcdam
    # figure to a granularity of 0.01. This can be observed in the parses.
    my $hit = $db + int(int($roll20*int($wd*$bs_mult+0.5)/10+0.5) * int(100*
$pcdam)/100+0.5);

    # Min backstab after AA
    $hit = 290 if ($hit < 290);
    $hist{$hit}++;
}

foreach (sort {$a <=> $b} keys(%hist)) {
    print "$_\t$hist{$_}\n";
}
```

Clearly the rounding has some work to do and distributions are complete guess work, but the basics seem to work.