One German Tank

The following problem was posed to me as a demonstration that Bayesian inference is not the best approach for all problems. It is a special case of the German Tank Problem (https://en.wikipedia.org/wiki/German_tank_problem). The name commemorates a famous statistical inference of the production of German tanks in World War II, based solely on the serial numbers recorded from a relatively small sample of tanks captured and inspected by the Allies. The special case of looking at only one serial number is equivalent to the following problem.

Problem: An unknown number of paper tickets, numbered $1, \ldots, N$, are placed into a hat. Ticket number m is drawn from the hat. Estimate N.

It is not obvious that a single draw gives enough information to estimate N. As we will see, the most straightforward Bayesian answer is that there is no answer. That should leave us a bit suspicious of the problem. Yet there is a simple *frequentist* answer: $m \in 1, \ldots, N$, with uniform probability $\Pr(m|N) = 1/N$ for each of these possibilities. Pretend we know N. Then the mean value of the ticket drawn is $\langle m \rangle = (N+1)/2$. Our frequentist estimate of N is then

$$\hat{N}_f = 2m - 1.$$

This makes intuitive sense, but the assumptions are perhaps a bit murky.

Bayesian Approach: Flat Prior

As already stated,

$$\Pr(m|N=n) = \begin{cases} \frac{1}{n}, & 1 \le m \le n, \\ 0, & m > n. \end{cases}$$

The posterior probability is

$$\Pr(N=n|m) = \frac{\Pr(m|N=n)\ \Pr(N=n)}{\Pr(m)}, \quad m \le N.$$

Marginalizing over all possible N,

$$\Pr(m) = \sum_{n'=m}^{\infty} \Pr(m|N=n')\ \Pr(N=n').$$

A flat prior, $\Pr(N=n) \propto 1$, cannot be normalized over its semi-infinite domain $N \in [1, \infty)$. Such a distribution is termed *improper*. Since the normalization cancels anyway, let's not worry about it. The posterior becomes

$$\Pr(N=n|m) = \frac{n^{-1}}{\sum_{n'=m}^{\infty} n'^{-1}}.$$

Since this is not normalizable, the median and mean do not exist. The best we can do is the (obviously biased) maximum likelihood estimate, $\hat{N}_{ML} = m$.

Modified Prior

As I see it, the only way for Bayesian inference to arrive at a useful answer is to posit a different prior, $\Pr(N = n)$. If pressed, I suppose most people would admit that infinite N, which is the source of all the difficulty, doesn't make much sense. The hat must have some finite capacity, even if I do not know what it is. In some sense, I am looking for the *flattest useful prior*, one that does not impose a particular scale on N before ticket number m is introduced. I propose:

$$\Pr(N = n) \propto n^{-\alpha}, \quad \alpha \geq 0. \quad (1)$$

Thus, the marginalization over N looks like

$$\Pr(m) = \frac{1}{\zeta(1+\alpha)} \sum_{n=m}^{\infty} n^{-(1+\alpha)},$$

where the normalization factor is given by the reciprocal of the Riemann zeta function,

$$\zeta(z) \equiv \sum_{k=1}^{\infty} k^{-z}.$$

which is undefined for $z = 1$ (corresponding to $\alpha = 0$, the flat prior). The posterior is then

$$\Pr(N = n|m) = \frac{n^{-(1+\alpha)}}{\sum_{n'=m}^{\infty} n'^{-(1+\alpha)}}. \quad (2)$$

Estimating N requires some chosen measure of the central tendency, $\hat{N}$, of this distribution. I propose that α should be an integer, small enough that the central tendency of the prior alone yields an undefined result, but large enough to provide sufficient information to obtain $\hat{N}$. Under this rule, *the scale is imposed by the evidence, not solely by the prior*. Specifically,

Measure of $\hat{N}$	α	$\hat{N}$ (nearest integer)	Comment
Mode	0	m	See previous section.
Median	1	$2m - \frac{1}{2}$	See below.
Mean	2	$2m - 1$	Matches $\hat{N}_f$. Derived below.

In the next sections I derive the results for the median and mean.

The Median

Exercise left to the reader.

The Mean

For the mean, $\hat{N} = \langle N \rangle$, my rule dictates that the prior goes as

$$\Pr(N = n) \propto n^{-2},$$

a distribution whose mean diverges logarithmically. The posterior is then

$$\Pr(N = n|m) \propto n^{-3}, \quad N \geq m.$$

Under the assumed prior, our estimate of N is then

$$\langle N \rangle = \frac{\sum_{n=m}^{\infty} n^{-2}}{\sum_{n=m}^{\infty} n^{-3}} = \frac{\zeta(2) - \sum_{n=1}^{m-1} n^{-2}}{\zeta(3) - \sum_{n=1}^{m-1} n^{-3}}.$$

Using the midpoint method approimation to the sums,

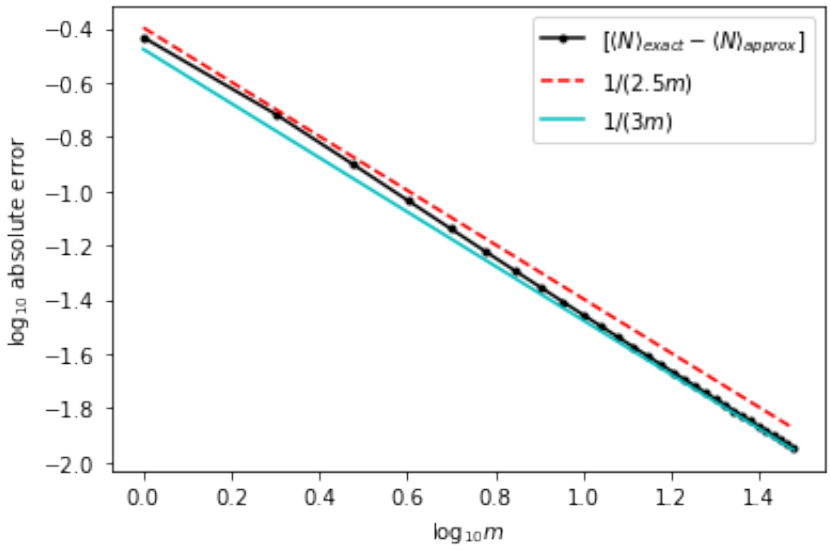
$$\langle N \rangle \approx \frac{\int_{m-1/2}^{\infty} x^{-2}\, dx}{\int_{m-1/2}^{\infty} x^{-3}\, dx} = 2m - 1.$$

The approximate formula always gives the value of the integer nearest to $\langle N \rangle$, which is all the accuracy we could reasonably desire, considering that $N \in 1, 2, 3, \ldots$. As the numerical test below demonstrates, the *absolute* error (not just fractional error) decays steadily with increasing m, always falling between the two plotted trend lines.

Oddly, the approximation happens to match the frequentist result, $\hat{N}_f = 2m - 1$.

In [1]:
```
# Numerical test of the approximate formula for the mean of N, wit
import numpy as np
import matplotlib.pyplot as plt
import scipy.special as sp

M = 30   # Maximum value of m for numerical test
m     = np.arange(1.0,M+1)
Napprox = 2*m - 1     # Approximate formula for <N>
numer = np.empty(M)   # Numerator of <N>
denom = np.empty(M)   # Denominator of <N>
numer[0] = sp.zeta(2)
denom[0] = sp.zeta(3)
for i in range(1,M):
    numer[i] = numer[i-1] - 1/i**(2)
    denom[i] = denom[i-1] - 1/i**(3)
Nexact = numer/denom  # Exact (to numerical precision) value of <N
print(Nexact[0])
plt.figure()
plt.plot(np.log10(m), np.log10(Nexact - Napprox),'k.-',
         label=r'$\left[ \left< N \right>_{exact} - \left< N \righ
plt.plot( np.log10(m), np.log10(1/(2.5*m)),'r--', label='$1/(2.5 m
plt.plot( np.log10(m), np.log10(1/(3*m)),'c-', label='$1/(3 m)$' )
plt.xlabel(r'$\log_{10} m$')
plt.ylabel(r'$\log_{10}$ absolute error')
plt.legend()
plt.show();
```

```
1.3684327776202059
```



Discussion

What have I learned from this exercise? Perhaps that I should have stuck with the frequentist approach, because I managed to get (approximately) the same result from Bayesian inference? But what does the frequentist result mean? The frequentist procedure was straightforward, but lacked the clarity of assumptions afforded by Bayesian inference.

I had to make a very particular assumption about the prior to get $\langle N \rangle = 2m - 1$ from Bayes' theorem. I did my best to justify that assumption, but there is nothing unique about my choices. For example, I could have picked any $\alpha \in (1, 2]$ to calculate $\langle N \rangle$. Any choice in that semi-open interval leads to a well-defined mean. At the same time, this range of choices does not directly impose a scale on the prior distribution, in that the mean of the prior diverges. Consider $m = 1$, which leads to

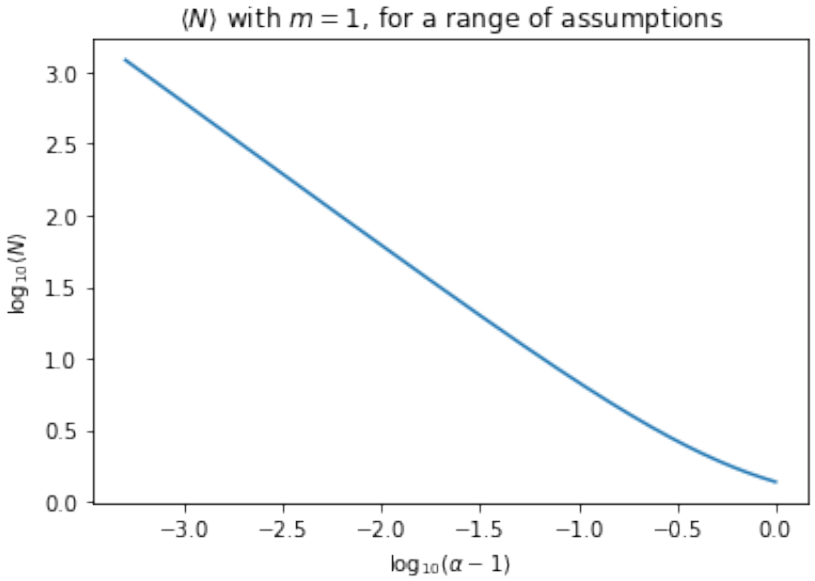
$$\langle N \rangle = \frac{\zeta(\alpha)}{\zeta(1 + \alpha)}.$$

As you can see from the plot below, this results in practically any value of $\langle N \rangle$ I might want. $\alpha = 2$ yields $\langle N \rangle = 1.36843$, a lower limit of what can be obtained for $\langle N \rangle$, but there is no upper limit. Likewise, $\langle N \rangle \to 2m - 1$ is a lower limit of what can be obtained from equation (1) for $m \gg 1$, but there is no upper limit.

Bayes' theorem can make very little sense of the frequentist result. Further below, I have included an Appendix with a different frequentist approach that yields a radically different answer. Perhaps the undefined result of the Bayesian analysis should have warned me that I was attempting to answer an *ill-posed question*.

I conclude that if we draw only one serial number m, all we can say is that $N \geq m$, and of any two particular choices, the smaller value of N is more probable.

In [2]:
```
plt.figure()
alpha = np.arange(1.0005,2,.001)
plt.plot(np.log10(alpha-1), np.log10( sp.zeta(alpha)/sp.zeta(1+alp
plt.xlabel(r'$\log_{10}(\alpha-1)$')
plt.ylabel(r'$\log_{10}\left<N\right>$')
plt.title(r'$\left<N\right>$ with $m=1$, for a range of assumptior
plt.show();
```



Appendix: Alternative Frequentist Solution

This was fun, but in the end it does not produce much insight. I may eventually delete it.

Now I pose the question: By what factor should I multiply m to get an estimate of N? It seems then that the starting point is not $\langle m \rangle$, but $\langle N/m \rangle$.

Pretend for a moment that we know N.

$$\left\langle \frac{N}{m} \right\rangle = \sum_{m'} \frac{N}{m'} \Pr(m'|N).$$

But recall

$$\Pr(m|N) = \begin{cases} \frac{1}{N}, & m \leq N \\ 0, & m > N \end{cases}$$

Hence,

$$\left\langle \frac{N}{m} \right\rangle = \sum_{m'=1}^{N} \frac{1}{m'}.$$

Of course, we don't really know N! But perhaps it would be reasonable to place $\langle N \rangle$ in the upper limit of the sum on the right hand side.* Since we do know m, I now propose

$$m \left\langle \frac{N}{m} \right\rangle = \boxed{\langle N \rangle = m \sum_{m'=1}^{\langle N \rangle} \frac{1}{m'}.}$$

The above equation cannot be solved for $\langle N \rangle$, but can be trivially solved for m. The only disadvantage is that the results are then for non-integer m (except $m = 1$ at $\langle N \rangle = 1$). A reasonable approximation can be made by turning the series into an integral using the midpoint method,

$$\langle N \rangle = m \left[1 + \log\left(\frac{2 \langle N \rangle + 1}{3} \right) \right].$$

This is transcendental, but the approximation is nice because it happens to be exact for $\langle N \rangle = 1$, very accurate for $\langle N \rangle \gg 1$, and $\langle N \rangle$ can be obtained quickly for any chosen m using, e.g., Newton's method.

The eyeball fit below is good enough for $m < 10000$:

$$\boxed{\langle N \rangle = 2m - 1 + 1.1 m \log m}$$

* Should I instead average the sum over all possible N?

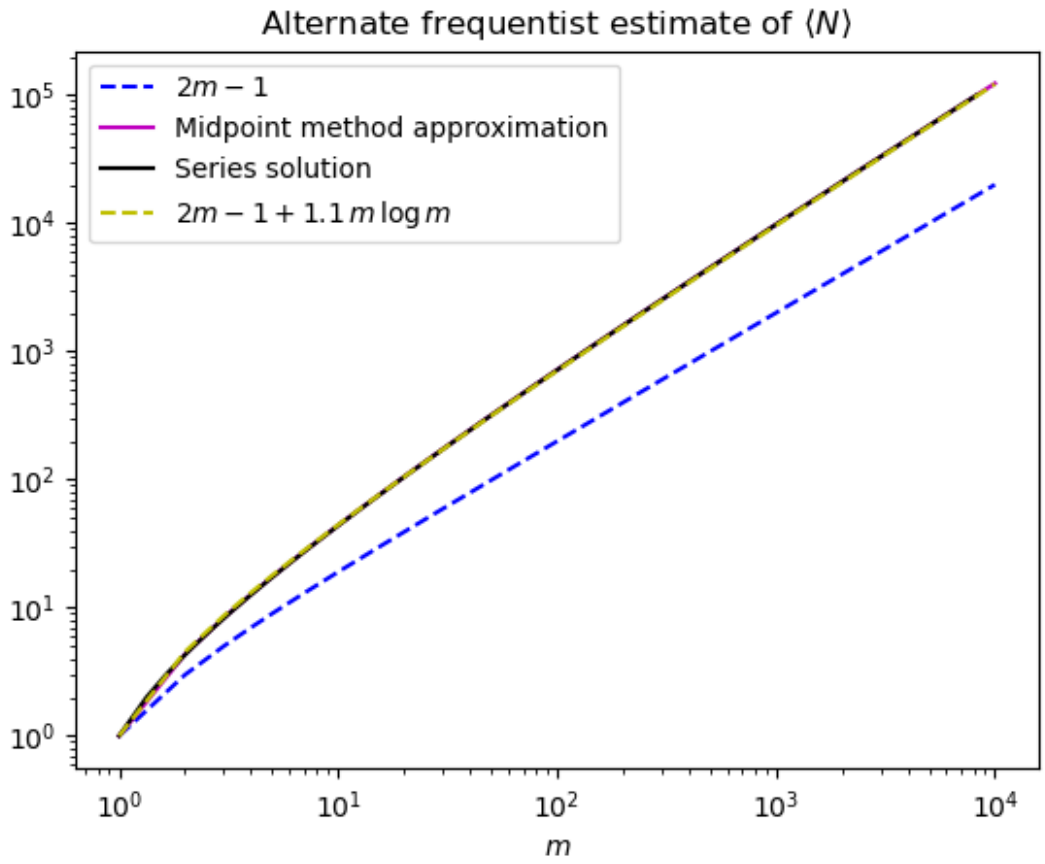
In [3]:
```
# Tabulating non-integer m from the series
Nmax = int(1e5)
N_array = np.arange(1,Nmax+1)
m_array = np.empty((Nmax))
harmonic_sum = 0
for i in range(Nmax):
    harmonic_sum += 1/N_array[i]
    m_array[i] = N_array[i]/harmonic_sum
```

In [4]:
```
# Root finding from the midpoint approximation....
m = 1  # Initialize m before defining the function
def rootme(N):
    return (  N - m*(1+np.log((2*N+1)/3)) )
```

In [5]:
```
from scipy.optimize import root
from scipy.optimize import newton
from time import perf_counter
M = 10000
ms = np.arange(1,M+1)
N_guess = 2*ms - 1  # Initial guess will use old frequentist estim
N_avg = np.empty(M)
t1 = perf_counter()
for i in range(M):
    m = ms[i]
    soln = root(rootme, N_guess[i])
    N_avg[i] = soln.x
t2 = perf_counter()
print('Execution Time = ',t2-t1,' seconds.')
```

```
Execution Time =  0.9318995069999998  seconds.
```

In [6]:
```
%matplotlib notebook
plt.figure()
plt.loglog(ms, N_guess, 'b--', label='$2m-1$')
plt.plot(ms, N_avg, 'm-', label='Midpoint method approximation')
plt.plot(m_array, N_array, 'k-', label='Series solution')
plt.plot(ms, N_guess + 1.1*ms*np.log(ms), 'y--',label='$2m - 1 + 1
plt.xlabel('$m$')
plt.title(r'Alternate frequentist estimate of $\left<N\right>$')
plt.legend()
plt.show();
```



Appendix: Debunking the Frequentist Solution

At the start of this notebook, I proposed a “frequentist” solution

$$\hat{N}_f = 2m - 1.$$

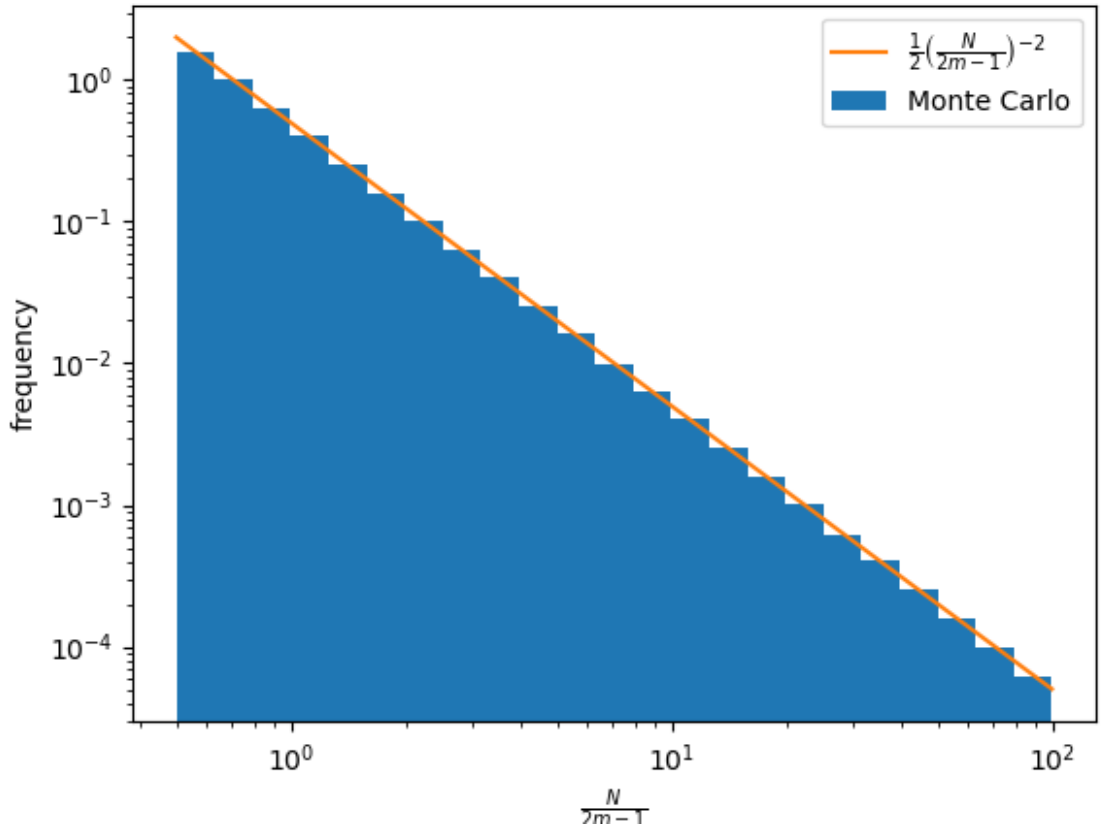
I will now elucidate the real significance of that solution by a combined of analytic and numerical approach. First, let's do a MonteCarlo calculation....

In [7]:
```
# Simplest test: Fixed N
N = 999
Ntries = 10000000
m_arr = np.random.randint(1, N+1, Ntries)
N_freq = 2*m_arr - 1  # Naive frequentist estimate of N. Always ar
print('True value of N:', N)
print(' Median of 2m-1:', np.median(N_freq))
print('   Mean of 2m-1:', np.mean(N_freq))
```

```
True value of N: 999
 Median of 2m-1: 999.0
   Mean of 2m-1: 998.6834486
```

In [8]:
```
Nmin = 1  #2**63-1024
Nmax = 2**10 # At most 2**63-2. Max int64 is 2**63-1.
Ntries = 1000000
N_arr = np.random.randint(Nmin, Nmax+1, Ntries)
# print(Nmax-Nmin)
m_arr = np.empty((Ntries))
for i in range(Ntries):
    # print(N_arr[i]+1)
    m_arr[i] = np.random.randint(1, N_arr[i]+1)
N_freq = 2*m_arr - 1  # Naive frequentist estimate of N.

quartiles = np.quantile(N_arr/N_freq, (0.25, 0.5, 0.75))

print('Number of MonteCarlo trials: ', Ntries)
print(' Maximum allowed value of N: ', Nmax)
print('              log10 of max N: ', np.log10(Nmax), '\n')
print(' First quartile of N/(2m-1): ', quartiles[0])
print('         Median of N/(2m-1): ', quartiles[1])
print(' Third quartile of N/(2m-1): ', quartiles[2], '\n')
print('           Mean of N/(2m-1): ', np.mean(N_arr/N_freq))
print('        Std dev of N/(2m-1): ', np.std(N_arr/N_freq))

plt.figure()
rmin = 0.5
rmax = 100
rarray = 10**np.arange(np.log10(rmin), np.log10(rmax), 0.1)
plt.hist(N_arr/N_freq, bins=rarray, range=[rmin,rmax], density=Tru

model = 0.5*rarray**-2
plt.plot(rarray,model,label=r'$\frac{1}{2}\left(\frac{N}{2m-1}\rig
plt.xlabel(r'$\frac{N}{2m-1}$')
plt.ylabel('frequency')
plt.xscale('log')
plt.yscale('log')
plt.legend()
plt.show()
```

```
Number of MonteCarlo trials:  1000000
 Maximum allowed value of N:  1024
             log10 of max N:  3.010299956639812

 First quartile of N/(2m-1):  0.6666666666666666
        Median of N/(2m-1):  1.0
 Third quartile of N/(2m-1):  2.0

           Mean of N/(2m-1):  3.9261925179678756
         Std dev of N/(2m-1):  24.606176290770858
```

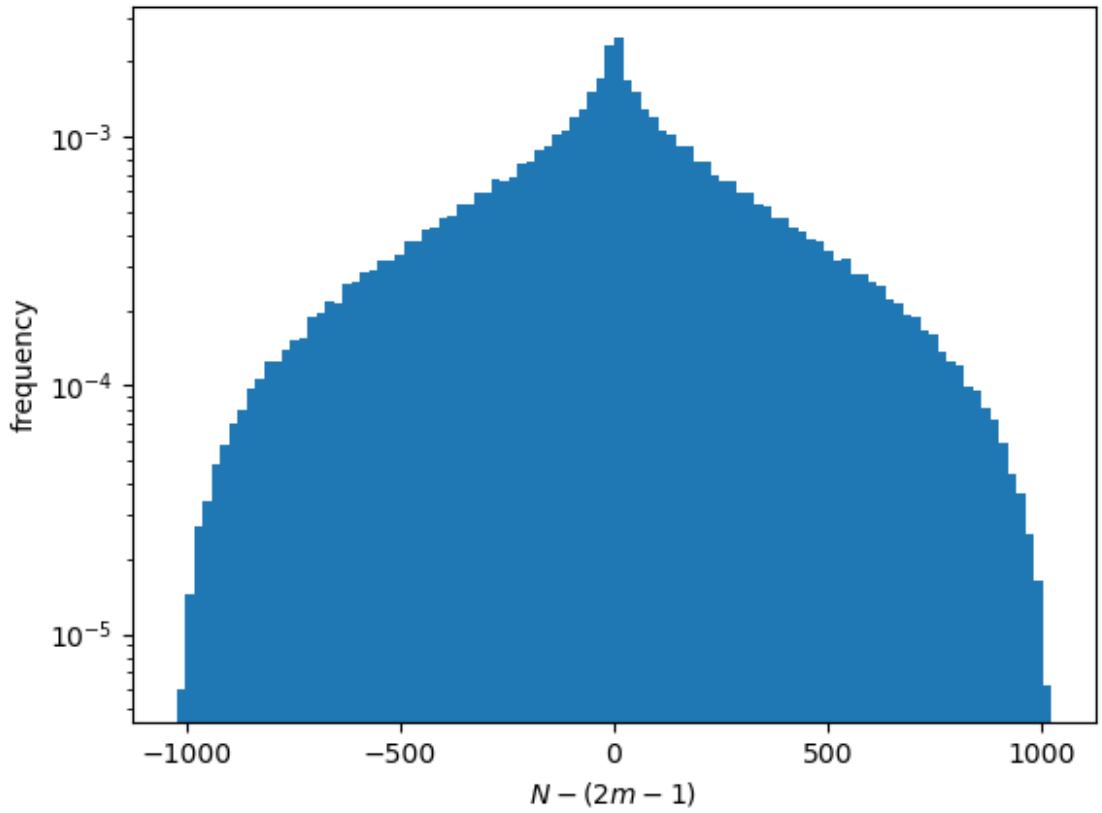


In [9]:

```
quartiles = np.quantile(N_arr - N_freq, (0.25, 0.5, 0.75))

print('Number of MonteCarlo trials: ', Ntries)
print(' Maximum allowed value of N: ', Nmax)
print('             log10 of max N: ', np.log10(Nmax), '\n')
print(' First quartile of N - (2m-1): ', quartiles[0])
print('         Median of N - (2m-1): ', quartiles[1])
print(' Third quartile of N - (2m-1): ', quartiles[2], '\n')
print('           Mean of N - (2m-1): ', np.mean(N_arr - N_freq))
print('        Std dev of N - (2m-1): ', np.std(N_arr - N_freq))

plt.figure()
```

```
plt.hist(N_arr - N_freq, bins=100, density=True)

# model = 0.5*rarray**-2
# plt.plot(rarray,model,label=r'$\left(\frac{N}{2m-1}\right)^{-2}$
plt.xlabel(r'$N - (2m-1)$')
plt.ylabel('frequency')
# plt.xscale('log')
plt.yscale('log')
plt.show()
```

```
Number of MonteCarlo trials:  1000000
 Maximum allowed value of N:  1024
            log10 of max N:  3.010299956639812

 First quartile of N - (2m-1):  -191.0
        Median of N - (2m-1):  0.0
 Third quartile of N - (2m-1):  191.0

           Mean of N - (2m-1):  -0.108432
         Std dev of N - (2m-1):  341.69206005481226
```



Result of the MoteCarlo

At this point, $\hat{N}_f = 2m - 1$ is looking very interesting. Averaging over an ensemble of N and m values, I have found that:

$$\left\langle \frac{N}{2m-1} \right\rangle_{\text{median}} = 1,$$

and

$$\langle N - (2m-1) \rangle_{\text{median}} = \langle N - (2m-1) \rangle_{\text{mean}} = 0.$$

Doesn't this mean that $2m - 1$ is a good estimate for N? The answer, unfortunately, is no, as I will demonstrate below.

The meaning of $2m-1$

When I calculated the means and medians above, I included all N and m. For comparison, then, I should calculate the median value of the distribution $\Pr(N-(2m-1))$. It is easiest to start by pretending to known N:

$$\Pr(N-(2m-1)|N) = \begin{cases} \frac{1}{N}, & m \in 1,\ldots,N; \\ 0, & \text{else.} \end{cases}$$

Now,

$$\Pr(N-(2m-1)) = \sum_N \Pr(N-(2m-1)|N)\ \Pr(N).$$

I want the median of the above distribution, but it turns out there is a slippery way to get it without doing the marginalization over N. The reason will become apparent if we take the median of $\Pr(N-(2m-1)|N)$. Since m takes on values $1,\ldots,N$ with equal probability, $2m-1 \in 1,3,5,\ldots,2N-1$. The median (and mean) value of $2m-1$ is thus N. Consequently, the median (and mean) of $N-(2m-1)$ is 0 when N is known. As this is independent of N, so the marginalization over N is not needed. The result is that

$$\langle N-(2m-1)\rangle_{\text{median}} = \langle N-(2m-1)\rangle_{\text{mean}} = 0.$$

This is what we found with our Monte Carlo calculation, and the neat thing is that it's independent of our prior, $\Pr(N)$.

It is tempting at this point to think that $2m-1$ is a good estimate of N. The appearance of m in this simple expression encourages us to think that we are taking m properly into account when we calculate m, but we are not! What we really want is the median of the distribution $\Pr(N-(2m-1)|m)$. Suppose we try to evaluate as follows:

$$\Pr(N-(2m-1)|m) = \frac{\Pr(N-(2m-1)|N)\ \Pr(N)}{\Pr(m)}.$$

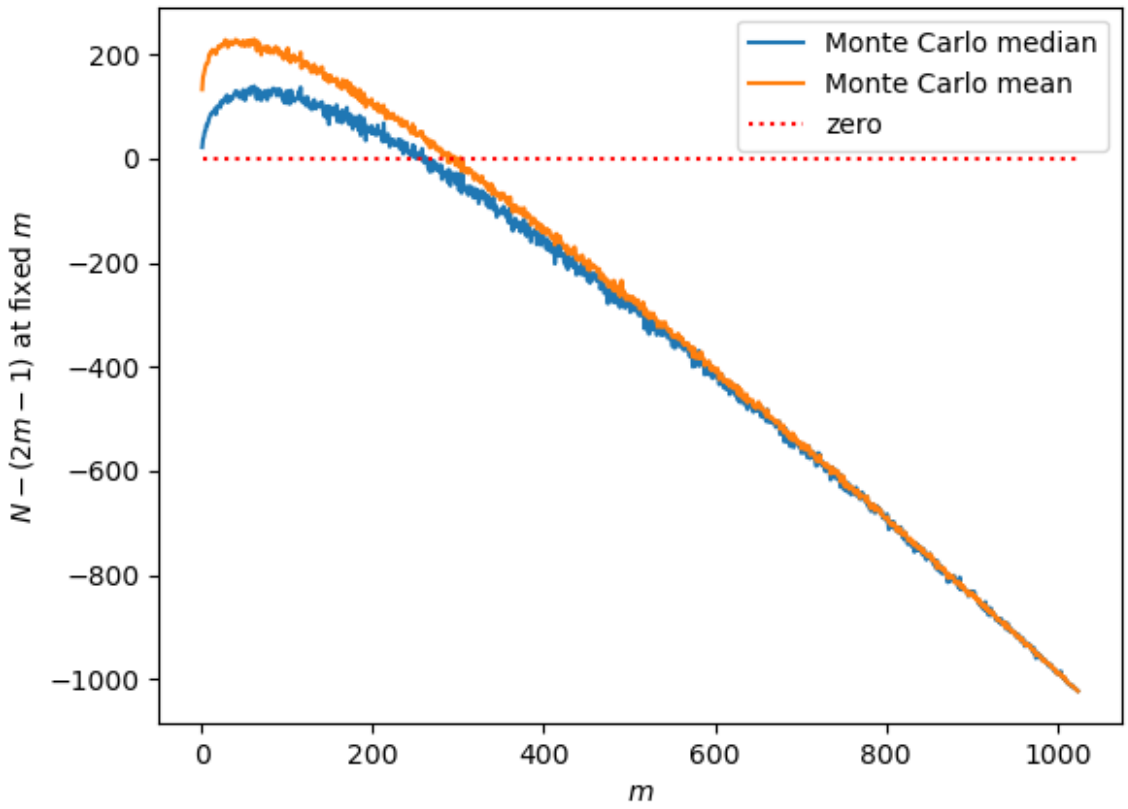
The denominator requires marginalizing over N:

$$\Pr(m) = \sum_{N\geq m} \Pr(N-(2m-1)|N)\ \Pr(N),$$

which introduces a dependence on m that will change depending on the form of $\Pr(N)$.

Below, I will demonstrate via MonteCarlo that $2m-1$ has no nice relationship to N when m is held fixed.

In [10]:
```
mmax = min((Nmax, 1024))
Nmed = np.empty((mmax))
Nmean = np.empty((mmax))
mm = np.arange(1,mmax+1)
for i in range(mmax):
    ss = np.where(m_arr == mm[i])
    Nmed[i] = np.median(N_arr[ss])
    Nmean[i] = np.mean(N_arr[ss])

plt.figure()
plt.plot(mm, Nmed-(2*mm-1), label=r'Monte Carlo median')
plt.plot(mm, Nmean-(2*mm-1), label=r'Monte Carlo mean')
plt.plot(mm, np.zeros(mm.shape),'r:', label=r'zero')
plt.xlabel(r'$m$')
plt.ylabel(r'$N-(2m-1)$ at fixed $m$')
plt.legend()
plt.show();
```



In [11]:

```
plt.figure()
plt.plot(mm, Nmed, label=r'Monte Carlo')
plt.plot(mm, 2*mm-1, label=r'$2m-1$')
plt.xlabel(r'$m$')
plt.ylabel(r'median $N$ at fixed $m$')
plt.legend()
plt.show();
```