

10 1.2 The only new objection that I can think of to the question of 'Can machines think?' is the possible objection that humans don't really think either and that both people and machines can be very complex machines taking a complex set of inputs, modifying themselves and creating an output. People and machines may have the appearance of thought, but the argument could be made that they are only processing an input and creating an output. Any other new objection that I thought of was a variation of one of the objections that Turing brought up or was invalid. Either way it seems that the question is somewhat poorly defined in regards to what it means 'to think'.

10 1.3 The most recent Loebner prize was Robert McDeksa who won for the Ultra Hal Assistant, a digital secretary that learns through interaction and remembers information that the user wants it to save.

10 Ultra Hal uses language processing in order to have natural conversations with the user. Ultra Hal advances the state of the art by keeping information the user inputs, learning voice recognition and learning about the user. It also interacts with other computer programs in order to store information as well as to help research different topics or access the web.

10 1.4 The fact that there are certain problems which are currently difficult or impossible does not mean that AI is impossible. It simply means that it has not been done yet, not that it can't be done. A new system or approach may be necessary. Another view is that there are things that some humans cannot do, but that does not mean that humans are necessarily unintelligent.

10 1.10 Yes, reflex reactions are rational. Reflex reactions are a learned feed-forward response. Reflex reactions make sense in that if a person touches a hot stove and did not pull away they would burn themselves. The response of pulling ones hand away is a subconscious thought. One does not usually think through in their head that 'this is hot, if I leave my hand here I will burn my hand, I had better pull my hand away.' It is not clear whether reflex reactions are intelligent. It depends on the definition of intelligent used here. If intelligence simply has to do with one's ability to learn, then yes reflex reactions are intelligent as they are a learned response. Animals seem to exhibit reflex responses but depending on whether or not animals are viewed as intelligent this may or may not have any effect on whether or not they are intelligent.

10 1.11 Depending on how this statement is viewed, this statement can be true. In one sense computers can do more than originally programmed due to learning algorithms. At the same time it can be argued that the computer has been programmed to learn, so it is doing what it was programmed to do.

If computers only do what they are programmed to do, that does not necessarily mean that they are not intelligent. Depending on how one views people and animals, then the

same argument could be applied. If one ignores the possibility of the soul and assumes that people have no free will, then a human would only be doing what they are programmed to do, so a computer would be no more or less intelligent than a human.

20  
1.6 The model is attached as the m file addition.m. Also attached are a few examples of the program being run and the times that it takes the program to run each time. The program was run without the time delay 10,000 times and a histogram was made of the distribution of the systems answers.

This could be modified to be used with multiplication by first altering the times and error rates so that they are not based on the sum, but rather on the multiplication. The types of errors that occur would have to be changed as well, some would be additions or subtractions to account for incorrect addition when adding the multiples together, and then some numbers could be added or subtracted prior to the multiplication.

1.7 The code for the following information is attached at the end of the packet as well as the command window information

30  
a.1) First set of thirty flips

HTTTTHTTTHTHTTTTHTHTTTTHTTTHTT

b.1) 30% heads

c.1) PHH=0; PTH=0.3571; PHT=1; PTT=0.6429

f.1) Last 10 flips based only on 21<sup>st</sup> flip 80% accurate

THTTHTHTHTT

TTTTHTTTHTT (actual)

Last 10 values based on actual flip each time 80% accurate

TTTTHTTTHTT

TTTTHTTTHTT (actual)

a.2) Due to the strange number of tails in the first trial, I did another set of 30.

HTTHHTHHHTHTHTTTTHHTHHHTTHTTHTHHHT

b.2) 55% heads

c.2) PHH=0.3636; PTH=0.6667; PHT=0.6364; PTT=0.3333

f.2) Last 10 flips based only on the 21<sup>st</sup> flip. 70% accurate

TTHTHHHTHTTT

TTHTTHTHHHT (actual)

Last 10 values based on the actual flip each time. 50% accurate

THHTTTTTTTT

TTHTTHTHHHT (actual)

Part d)

a.3) Other persons 'random' choices of heads and tails.

HHTHTHHHTHTHTTTHTHTTTHHTHTTTHTHT

b.3) 50% heads

c.3) PHH=30%; PTH=70%; PHT=70%; PTT=30%

f.3) Last 10 flips based only on 21<sup>st</sup> flip 60% accurate

HTTTTHTHTHH

HTHTTTHTHT (actual)

Last 10 values based on actual flip each time 80% accurate

HTHTTTHTHT

HTHTTTHTHT (actual)

g) These results are sort of strange. While random, the first coin flip set was very strange and played to the program very well giving it a high accuracy. After redoing the coin flips, the 50% accuracy makes sense, but the 70% is surprisingly high. It seems that it should be closer to 50%. The sequence chosen by the other person fit the pattern the most. It makes sense that the second method for determine the string would be higher because it compares itself to each actual coin flip while the other one only goes off of one flip. The issue with the first method is that if it guesses wrong early on, then the entire chain will be shifted making more guesses incorrect. When I asked the other person to write down their choice of coin flips, they commented at the end 'I realized I was following a pattern, so I changed it at the end' which made me nervous that it would completely change the values, but it seems to have been alright. In order to see how well this actually worked, it seems like the tests would need to be repeated more times.

```

function ans = addition(a,b)

prob=rand(1);
%time delays are partially random and partially dependent on the sum of the
%numbers. the time delay is aimed more at an average person rather than a
%room of mathematicians or engineers.
pause((prob/2+0.5)*(log((a+b+1))^1.2)+1);

small=min(a,b);
big=max(a,b);
ratio=small/big;

%order determination
o=a;
o1=0;
u=b;
o2=0;
while o>10
    o1=o1+1;
    o=o/10;
end
while u>10
    o2=o2+1;
    u=u/10;
end

orderratio=min(o1,o2)/max(o1,o2);
r=min(o1,o2);
s=max(o1,o2);
if (a+b)<10
    ans=a+b;
else
    mistakes=1;
    ans2b=a+b;%this sets it up so the next part can loop in the event of certain random
values, this enables multiple errors to occur in each addition
    if rand(1) < (orderratio/35 + (1-exp(-(a+b)/5000)^1/2))/35
        %the possibility of an error depends on the difference in order
        %between the two values and magnitude of the overall sum.
        %in the case of an error
        updown=round(rand(1))
        while mistakes>=rand(1)
            mistakes=mistakes*0.4;
            comp=round(rand(1)*10);
            if comp==1|0|10
                if updown==1
                    ans2b=ans2b+10^r;%this seems like the most likely simple mistake
                else
                    ans2b=ans2b-10^r;
                    %i considered separating this mistake into some others
                    %where random numbers were added or subtracted, but this
                    %seemed more likely
                end
            end
        end
    end
end

```

```
elseif comp==2
    if updown==1
        ans2b=ans2b+10^s;
    else
        ans2b=ans2b-10^s;
    end
elseif comp==3
    if updown==1
        ans2b=ans2b+10^r+10^s;
    else ans2b=ans2b+10^r-10^s;
    end
elseif comp==4%this could be combined into previous, each as 2|4 and 3|4
    if updown==1
        ans2b=ans2b+round(10^(r-1));
    else ans2b=ans2b-round(10^(r-1));
    end
elseif comp==5
    if updown==1
        ans2b=ans2b+round(10^(s-1));
    else ans2b=ans2b-round(10^(s-1));
    end
elseif comp==6
    if updown==1
        ans2b=ans2b+round(10^(r-2));
    else ans2b=ans2b-round(10^(r-2));
    end
elseif comp==7
    if updown==1
        ans2b=ans2b+round(10^(s-2));
    else ans2b=ans2b-round(10^(s-2));
    end
elseif comp==8
    if updown==1
        ans2b=ans2b+round(10^(r-3));
    else ans2b=ans2b-round(10^(r-3));
    end
elseif comp==9
    if updown==1
        ans2b=ans2b+round(10^(s-3));
    else ans2b=ans2b-round(10^(s-3));
    end
end
end
ans=ans2b;
else
    ans=a+b;
end
end
```

```
>> tic
>> addition(2342,98437)
```

```
ans =
```

```
100779
```

```
>> toc
Elapsed time is 26.472953 seconds.
```

```
>> tic
>> addition(4538,2938)
```

```
ans =
```

```
7476
```

```
>> toc
Elapsed time is 22.114131 seconds.
```

```
>> tic
>> addition(234,472)
```

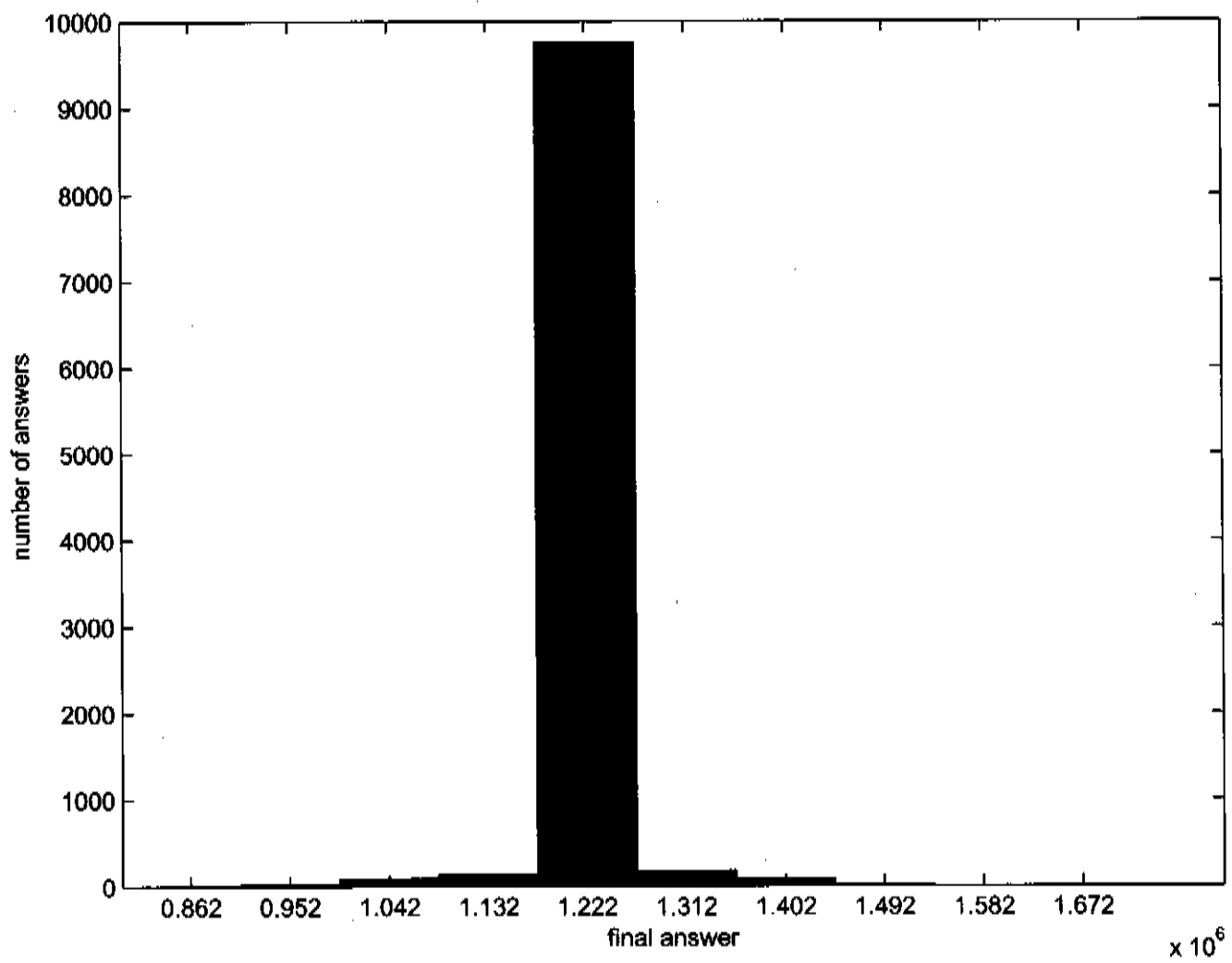
```
ans =
```

```
706
```

```
>> toc
Elapsed time is 19.204770 seconds.
```

```
>> %there is some extra error here because I am typing tic and toc
```

```
>> for i=1:10000
nums(end+1)=addition(234821,982174);
end
>> hist(nums)
>> xlabel('final answer')
>> ylabel('number of answers')
>>
```



```
function ans=probability(vec,num)
%vec needs to be input in a string 'HTHTHT'
%with single parentheses on each end
%num is the number of coins to look at
heads=0;

for i=1:num
    if vec(i)=='H'
        heads=heads+1;
    else
        end
end

ans=heads/num;
```

```

function [PHH PTH PHT PTT comparison comparison2] = markov(vec,num)
%where vec is the vector being observed and num is the number of points that you want to
look at in the vector to
%compute the markov transitions

HH=0;
TH=0;
HT=0;
TT=0;
%this looks at the current point and the next point to see how it would
%change, for the first 20 transitions, use num=20
for i=1:num
    if vec(i)=='H';
        if vec(i+1)=='H';
            HH=HH+1;
        elseif vec(i+1)=='T';
            HT=HT+1;
        end
    elseif vec(i)=='T';
        if vec(i+1)=='H';
            TH=TH+1;
        elseif vec(i+1)=='T';
            TT=TT+1;
        end
    end
end
%here are the percentages of each of these in groups
PHH=HH/(HH+HT);
PHT=HT/(HH+HT);
PTH=TH/(TH+TT);
PTT=TT/(TH+TT);

currentstate=vec(num+1);
comparison=[vec(num+1)];%this starts where the probabilities ended
for i=(num+1):length(vec)-1;%the -1 here is because we are starting at num+1 and we don't
need to guess for the final value of the chain
    odds=rand(1);
    if comparison(end)=='H';
        if odds<=PHH;
            comparison(end+1)='H';
        else comparison(end+1)='T';
        end
    elseif comparison(end)=='T';
        if odds<=PTT;
            comparison(end+1)='T';
        else comparison(end+1)='H';
        end
    end
end
comparison;

currentstate=vec(num+1);

```

```
comparison2=vec(num+1);%this starts where probabilities ends
for i=(num+1):length(vec)-1;
    odds=rand(1);
    if vec(i)=='H';
        if odds<=PHH;
            comparison2(end+1)='H';
        else comparison2(end+1)='T';
        end
    elseif vec(i)=='T';
        if odds<=PTT;
            comparison2(end+1)='T';
        else comparison2(end+1)='H';
        end
    end
end
comparison2;
```

The command window was put into word in order to save space

```
>> veca
```

```
veca =HTTTHHTHTHTHTHTTTTHHTHHTTTHTTHTHHHT
```

```
>> vecb
```

```
vecb =HTTTTHTTTHTHTTTTTHTHTTTTHTTTTHTT
```

```
>> vecn
```

```
vecn =HHTHTTTHHTHTHTTHTHTTTHHTTTHHTHT
```

```
>> probability(veca,20)
```

```
ans = 0.5500
```

```
>> probability(vecb,20)
```

```
ans = 0.3000
```

```
>> probability(vecn,20)
```

```
ans = 0.5000
```

```
>> [PHH PTH PHT PTT comparison comparison2] = markov(veca,20)
```

```
PHH = 0.3636
```

```
PTH = 0.6667
```

```
PHT = 0.6364
```

```
PTT = 0.3333
```

```
comparison = THTTHTHTHTT
```

```
comparison2 = TTTTHTHTTTT
```

```
>> [PHH PTH PHT PTT comparison comparison2] = markov(vecb,20)
```

```
PHH = 0
```

```
PTH = 0.3571
```

```
PHT = 1
```

```
PTT = 0.6429
```

```
comparison = TTHTHHTHTTT
```

```
comparison2 = THHTTTTTTTT
```

```
>> [PHH PTH PHT PTT comparison comparison2] = markov(vecn,20)
```

```
PHH = 0.3000
```

```
PTH = 0.7000
```

```
PHT = 0.7000
```

```
PTT = 0.3000
```

```
comparison = HTTTTHTHTHH
```

```
comparison2 = HTHTTHTTHTT
```