This is a sample project report for MATH 242. Take note of the style, organization, level of detail, and explanations presented below. Pay special attention to the items highlighted by purple comments and arrows.

Sample Project Report

Notebook is organized with a title, section, and subsection headings.

MATH 242 — Modern Computational Mathematics
Prof. Wright

Acknowledge resources you used, people you talked with, and ideas that are not your own.

Resource Acknowledgement: While working on this project, I referred to the textbook and to the Mathematica documentation. I briefly discussed my problem-solving approach with the professor in office hours.

Problem Statement

Include an introduction that summarizes the problem and states what you will do in the project.

I will use Madhava's formula to compute approximation of π:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ldots\right)$$

This formula can be obtained from the Maclaurin series for the arctangent function. It converges very slowly to π, meaning that lots of terms of the sum are required to obtain even a few digits of π. In class, we found that the first n terms of the sum produce about $\log_{10}(n)$ correct digits of π. In this project, I will determine exactly how many terms of the sum are required to obtain several correct digits of π. I will explore the sequence $n_1, n_2, n_3, \ldots$ where n_k is the minimum number of terms required to obtain k correct digits of π past the decimal point.

Implementation

Computing partial sums of Madhava's series

First, I use a loop with a Print statement to generate the terms of the sum. This is just a warm-up that helps me write the following code correctly.

In[1]:=
```
Do[Print[ (-1)^i / (2 i + 1) ], {i, 0, 4}]
```

$$1$$

$$-\frac{1}{3}$$

$$\frac{1}{5}$$

$$-\frac{1}{7}$$

$$\frac{1}{9}$$

Use text cells to explain what you will do in your code and what you observe from your output.

Seeing that the terms above are correct, I add them up and multiply by 4. I put this code in a module that accepts the number of terms *n* as an argument and returns the sum.

In[2]:=
```
computeSum[n_] := Module[{sum = 0},
  Do[sum += (-1)^i / (2 i + 1), {i, 0, n - 1}];
  4 * sum
 ]
```

Note that **computeSum** returns a fraction, which we can convert to a decimal number:

In[3]:=
```
computeSum[100]
```

Out[3]=
```
8 252 079 759 413 970 386 664 454 687 621 174 435 983 101 115 012 912 631 997 769 614 579 677 862 845⋱
    786 070 667 088 /
  2 635 106 162 757 236 442 495 826 303 084 698 495 565 581 115 509 040 892 412 867 358 728 390 766⋱
    099 042 109 898 375
```

In[4]:=
```
N[computeSum[1000], 50]
```

Out[4]=
```
3.1405926538397929259635965028693959704513893307797
```

The previous output is close to π, so it appears that our module **computeSum** is working as expected.

Counting correct digits

To find the number of correct digits past the decimal point, I use another module. This module also requires a parameter *n* that specifies the number of terms of the sum to use:

In[12]:=

```
correctDigits[n_] := Module[
  (* local variables within this module *)
  {approx, exact,
   approxDigit = 0, exactDigit = 0,
   count = 0},

  (* compute the partial-sum approximation with n terms;
  subtract 3 since we are interested in digits past the decimal point *)
  approx = computeSum[n] - 3;

  (* store the "exact" digits of pi past the decimal point *)
  exact = Pi - 3;

  (* use a loop to count correct digits *)
  While[approxDigit == exactDigit,
   (* multiply by 10 to shift the next digit to the left of the decimal point *)
   approx = 10 * approx;
   exact = 10 * exact;
   (* use the Floor function to peel off the next digit *)
   approxDigit = Floor[approx];
   exactDigit = Floor[exact];
   (* if the digits match, then increment the counter *)
   If[approxDigit == exactDigit,
    count += 1];
   (* print the current values for testing / debugging *)
   (*Print[{"approx digit",approxDigit,"exact digit",exactDigit,"count",count}];*)
   (* subtract the current digit so that we're ready to examine the next digit *)
   approx = approx - approxDigit;
   exact = exact - exactDigit;
  ];

  (* return the number of correct digits *)
  count
 ]
```

Use comments within larger blocks of code to explain what your code is doing.

I will check to make sure that **correctDigits** module works, using a **Print** statement to see the values of internal variables. The sum with 800 terms has two correct digits after the decimal:

In[7]:=

```
computeSum[800] // N
```

Out[7]=

```
3.14034
```

In[8]:=
```
correctDigits[800]
```
```
{approx digit, 1, exact digit, 1, count, 1}
{approx digit, 4, exact digit, 4, count, 2}
{approx digit, 0, exact digit, 1, count, 2}
```
Out[8]=
```
2
```

The sum with 5000 terms has three correct digits after the decimal:

In[9]:=
```
computeSum[5000] // N
```
Out[9]=
```
3.14139
```

In[10]:=
```
correctDigits[5000]
```
```
{approx digit, 1, exact digit, 1, count, 1}
{approx digit, 4, exact digit, 4, count, 2}
{approx digit, 1, exact digit, 1, count, 3}
{approx digit, 3, exact digit, 5, count, 3}
```
Out[10]=
```
3
```

Use printed output to demonstrate that your code works as expected. Once you are sure your code works, you may comment out Print statements within your module to avoid printing hundreds of lines of output.

It looks like the code is working properly, so now we will comment out the **Print** statement in the module.

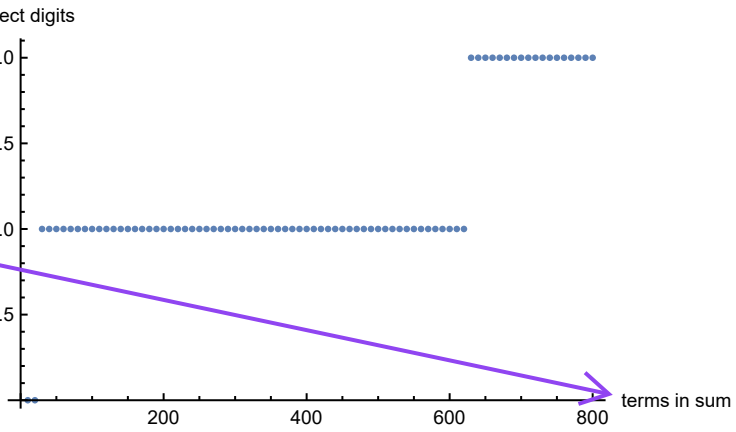
How many terms for each correct digit?

Now I will plot of the number of correct digits for various values of *n*. I will let *n* take values from 10 to 800, with a step size of 10.

In[13]:=
```
nDigits = Table[{n, correctDigits[n]}, {n, 10, 800, 10}]
```
Out[13]=
```
{{10, 0}, {20, 0}, {30, 1}, {40, 1}, {50, 1}, {60, 1}, {70, 1}, {80, 1}, {90, 1}, {100, 1},
  {110, 1}, {120, 1}, {130, 1}, {140, 1}, {150, 1}, {160, 1}, {170, 1}, {180, 1}, {190, 1},
  {200, 1}, {210, 1}, {220, 1}, {230, 1}, {240, 1}, {250, 1}, {260, 1}, {270, 1}, {280, 1},
  {290, 1}, {300, 1}, {310, 1}, {320, 1}, {330, 1}, {340, 1}, {350, 1}, {360, 1}, {370, 1},
  {380, 1}, {390, 1}, {400, 1}, {410, 1}, {420, 1}, {430, 1}, {440, 1}, {450, 1}, {460, 1},
  {470, 1}, {480, 1}, {490, 1}, {500, 1}, {510, 1}, {520, 1}, {530, 1}, {540, 1}, {550, 1},
  {560, 1}, {570, 1}, {580, 1}, {590, 1}, {600, 1}, {610, 1}, {620, 1}, {630, 2}, {640, 2},
  {650, 2}, {660, 2}, {670, 2}, {680, 2}, {690, 2}, {700, 2}, {710, 2}, {720, 2},
  {730, 2}, {740, 2}, {750, 2}, {760, 2}, {770, 2}, {780, 2}, {790, 2}, {800, 2}}
```

In[14]:=
```
ListPlot[nDigits, AxesLabel → {"terms in sum", "correct digits"}]
```

Out[14]=



Use plots to display numerical data as appropriate. Make sure your plots have proper labels.

We see the first correct digit is obtained with between 20 and 30 terms, and the second correct digit requires between 620 and 630 terms. Let's determine these value more precisely.

First correct digit

I will find the numbers of correct digits obtained by 20 to 30 terms.

In[15]:=
```
nDigits = Table[{n, correctDigits[n]}, {n, 20, 30}]
```

Out[15]=
```
{{20, 0}, {21, 1}, {22, 0}, {23, 1}, {24, 0}, {25, 1}, {26, 1}, {27, 1}, {28, 1}, {29, 1}, {30, 1}}
```

Fascinating! We see that 21 terms give the first correct digit past the decimal point, but then this digit is incorrect with 22 terms. This must be due to the fact that Madhava's series is an *alternating* series, with alternately positive and negative terms. With 25 or more terms, this digit seems to be consistently correct. Let's collect a bit more data to see if the first digit is ever correct with less than 20 terms.

In[16]:=
```
nDigits = Table[{n, correctDigits[n]}, {n, 10, 40}]
```

Out[16]=
```
{{10, 0}, {11, 0}, {12, 0}, {13, 0}, {14, 0}, {15, 0}, {16, 0}, {17, 0}, {18, 0}, {19, 1}, {20, 0},
  {21, 1}, {22, 0}, {23, 1}, {24, 0}, {25, 1}, {26, 1}, {27, 1}, {28, 1}, {29, 1}, {30, 1},
  {31, 1}, {32, 1}, {33, 1}, {34, 1}, {35, 1}, {36, 1}, {37, 1}, {38, 1}, {39, 1}, {40, 1}}
```

Typeset mathematical notation properly. Pay special attention to exponents, subscripts, and other symbols.

Now we see that the first decimal digit is correct with 19 terms, but it's not consistently correct until we have 25 or more terms. Since I think it's important for the digits to be consistently correct, I will set $n_1 = 25$. That is, it appears that $n_1 = 25$ is the smallest integer such that Madhava's series gives one correct digit of π (past the decimal point) with $n \geq n_1$ terms.

Second correct digit

I know that about 620 terms will be required to obtain the second digit of π. Since additional terms may be some alternating correct/incorrect, we will find the number of correct digits for n terms, for n in an interval around 620.

In[17]:=
```
nDigits = Table[{n, correctDigits[n]}, {n, 600, 650}]
```

Out[17]=
```
{{600, 1}, {601, 2}, {602, 1}, {603, 2}, {604, 1}, {605, 2}, {606, 1}, {607, 2}, {608, 1},
  {609, 2}, {610, 1}, {611, 2}, {612, 1}, {613, 2}, {614, 1}, {615, 2}, {616, 1}, {617, 2},
  {618, 1}, {619, 2}, {620, 1}, {621, 2}, {622, 1}, {623, 2}, {624, 1}, {625, 2}, {626, 1},
  {627, 2}, {628, 2}, {629, 2}, {630, 2}, {631, 2}, {632, 2}, {633, 2}, {634, 2},
  {635, 2}, {636, 2}, {637, 2}, {638, 2}, {639, 2}, {640, 2}, {641, 2}, {642, 2},
  {643, 2}, {644, 2}, {645, 2}, {646, 2}, {647, 2}, {648, 2}, {649, 2}, {650, 2}}
```

With a bit of trial and error, it seems that we get two correct digits of π when adding up 627 or more terms of Madhava's series. So let $n_2 = 627$.

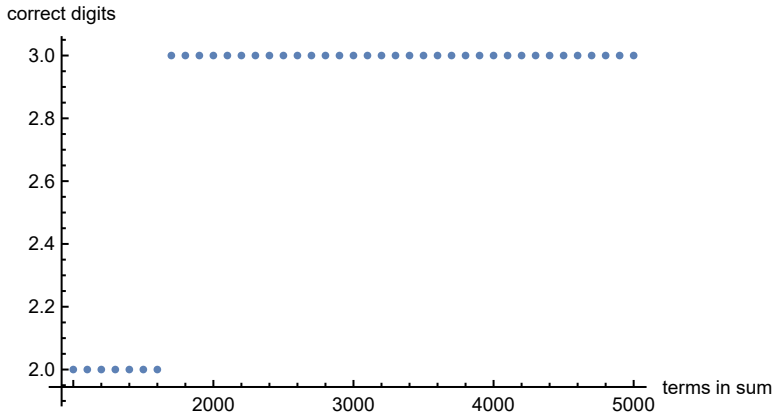
Third correct digit

I am guessing that the third correct digit will require a few thousand terms. Here is an initial calculation to determine this:

In[18]:=
```
nDigits = Table [ {n, correctDigits [ n ] }, {n, 1000, 5000, 100} ]
ListPlot [ nDigits, AxesLabel → {"terms in sum", "correct digits"} ]
```

Out[18]=
```
{ {1000, 2}, {1100, 2}, {1200, 2}, {1300, 2}, {1400, 2}, {1500, 2},
  {1600, 2}, {1700, 3}, {1800, 3}, {1900, 3}, {2000, 3}, {2100, 3}, {2200, 3},
  {2300, 3}, {2400, 3}, {2500, 3}, {2600, 3}, {2700, 3}, {2800, 3}, {2900, 3},
  {3000, 3}, {3100, 3}, {3200, 3}, {3300, 3}, {3400, 3}, {3500, 3}, {3600, 3},
  {3700, 3}, {3800, 3}, {3900, 3}, {4000, 3}, {4100, 3}, {4200, 3}, {4300, 3},
  {4400, 3}, {4500, 3}, {4600, 3}, {4700, 3}, {4800, 3}, {4900, 3}, {5000, 3} }
```

Out[19]=

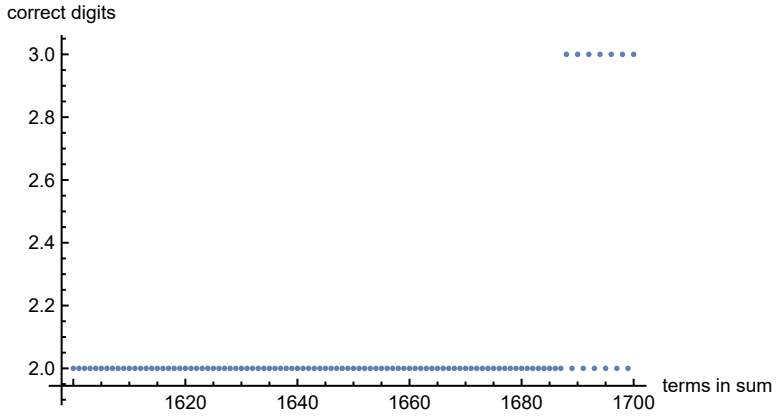


It turns out that the third correct digit is obtained somewhere between 1600 and 1700 terms. Let's see how many correct digits are obtained for each number of terms in this range.

In[20]:=
```
nDigits = Table [ {n, correctDigits [ n ] }, {n, 1600, 1700} ]
ListPlot [ nDigits, AxesLabel → {"terms in sum", "correct digits"} ]
```

Out[20]=
```
{{1600, 2}, {1601, 2}, {1602, 2}, {1603, 2}, {1604, 2}, {1605, 2}, {1606, 2}, {1607, 2},
  {1608, 2}, {1609, 2}, {1610, 2}, {1611, 2}, {1612, 2}, {1613, 2}, {1614, 2}, {1615, 2},
  {1616, 2}, {1617, 2}, {1618, 2}, {1619, 2}, {1620, 2}, {1621, 2}, {1622, 2}, {1623, 2},
  {1624, 2}, {1625, 2}, {1626, 2}, {1627, 2}, {1628, 2}, {1629, 2}, {1630, 2}, {1631, 2},
  {1632, 2}, {1633, 2}, {1634, 2}, {1635, 2}, {1636, 2}, {1637, 2}, {1638, 2}, {1639, 2},
  {1640, 2}, {1641, 2}, {1642, 2}, {1643, 2}, {1644, 2}, {1645, 2}, {1646, 2}, {1647, 2},
  {1648, 2}, {1649, 2}, {1650, 2}, {1651, 2}, {1652, 2}, {1653, 2}, {1654, 2}, {1655, 2},
  {1656, 2}, {1657, 2}, {1658, 2}, {1659, 2}, {1660, 2}, {1661, 2}, {1662, 2}, {1663, 2},
  {1664, 2}, {1665, 2}, {1666, 2}, {1667, 2}, {1668, 2}, {1669, 2}, {1670, 2}, {1671, 2},
  {1672, 2}, {1673, 2}, {1674, 2}, {1675, 2}, {1676, 2}, {1677, 2}, {1678, 2}, {1679, 2},
  {1680, 2}, {1681, 2}, {1682, 2}, {1683, 2}, {1684, 2}, {1685, 2}, {1686, 2},
  {1687, 2}, {1688, 3}, {1689, 2}, {1690, 3}, {1691, 2}, {1692, 3}, {1693, 2},
  {1694, 3}, {1695, 2}, {1696, 3}, {1697, 2}, {1698, 3}, {1699, 2}, {1700, 3}}
```

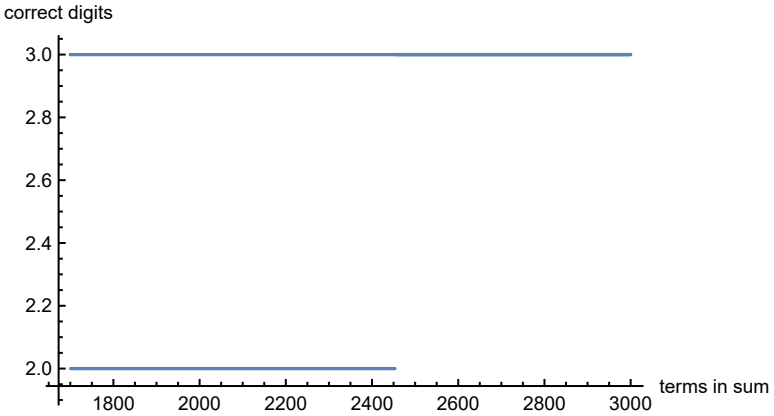
Out[21]=



We see that the third correct digit first appears for 1688 terms, but then this digit is alternately correct and incorrect as we add successive terms. Let’s explore what happens above 1700 terms. It seems we might have to go rather high, so we’ll focus on a plot first.

In[22]:=
```
nDigits = Table [ {n, correctDigits [ n ] }, {n, 1700, 3000} ] ;
ListPlot [ nDigits, AxesLabel → {"terms in sum", "correct digits"} ]
```

Out[23]=

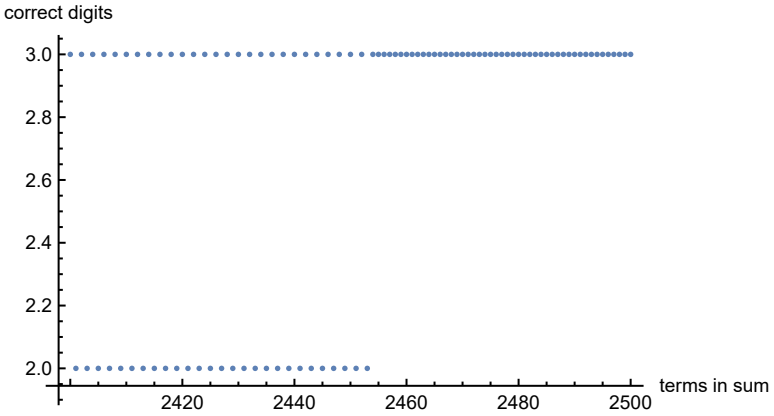


The last *n* that gives only two correct digits seems to be 2453, so let's zoom in near that value.

In[24]:=
```
nDigits = Table [ {n, correctDigits [n] }, {n, 2400, 2500} ]
ListPlot [ nDigits, AxesLabel → {"terms in sum", "correct digits"} ]
```

Out[24]=
```
{{2400, 3}, {2401, 2}, {2402, 3}, {2403, 2}, {2404, 3}, {2405, 2}, {2406, 3}, {2407, 2},
  {2408, 3}, {2409, 2}, {2410, 3}, {2411, 2}, {2412, 3}, {2413, 2}, {2414, 3}, {2415, 2},
  {2416, 3}, {2417, 2}, {2418, 3}, {2419, 2}, {2420, 3}, {2421, 2}, {2422, 3}, {2423, 2},
  {2424, 3}, {2425, 2}, {2426, 3}, {2427, 2}, {2428, 3}, {2429, 2}, {2430, 3}, {2431, 2},
  {2432, 3}, {2433, 2}, {2434, 3}, {2435, 2}, {2436, 3}, {2437, 2}, {2438, 3}, {2439, 2},
  {2440, 3}, {2441, 2}, {2442, 3}, {2443, 2}, {2444, 3}, {2445, 2}, {2446, 3}, {2447, 2},
  {2448, 3}, {2449, 2}, {2450, 3}, {2451, 2}, {2452, 3}, {2453, 2}, {2454, 3}, {2455, 3},
  {2456, 3}, {2457, 3}, {2458, 3}, {2459, 3}, {2460, 3}, {2461, 3}, {2462, 3}, {2463, 3},
  {2464, 3}, {2465, 3}, {2466, 3}, {2467, 3}, {2468, 3}, {2469, 3}, {2470, 3}, {2471, 3},
  {2472, 3}, {2473, 3}, {2474, 3}, {2475, 3}, {2476, 3}, {2477, 3}, {2478, 3}, {2479, 3},
  {2480, 3}, {2481, 3}, {2482, 3}, {2483, 3}, {2484, 3}, {2485, 3}, {2486, 3},
  {2487, 3}, {2488, 3}, {2489, 3}, {2490, 3}, {2491, 3}, {2492, 3}, {2493, 3},
  {2494, 3}, {2495, 3}, {2496, 3}, {2497, 3}, {2498, 3}, {2499, 3}, {2500, 3}}
```

Out[25]=



It looks like we consistently get three correct digits for $n \geq 2454$, so we have $n_3 = 2454$.

Discussion

Conclusions and Conjectures

Summarize your observations and conclusions at the end of your notebook.

As shown in the computations, plots, and discussion above, we found $n_1 = 25$, $n_2 = 627$, and $n_3 = 2454$. That is, when $n \geq n_k$, the sum of the first n terms of Madhava series produces at least k correct digits of π past the decimal point. We know from class that the Madhava series requires about ten times as many terms to produce each additional correct digit of π. Thus, it makes sense that $n_2 \geq 10\, n_1$ and $n_3 \approx 100\, n_1$. I would guess that n_4 is about 25,000, but it could be substantially larger than this.

Limitations and Extensions

Consider what limitations are present in your code and methodology.

My method of running **computeSum[n]** for various value of n is not a great way of figuring out how many terms are required to obtain a desired number of digits of π. As n gets larger, it gets more difficult to figure out the next terms in the sequence. I could write more efficient code that would better automate this search. Perhaps this could be also studied theoretically (rather than computationally) using results about rates of convergence of infinite series.

Consider directions for future study.

I am curious about the alternating behavior that I notice near each sequence value n_k. Specifically, I would like to better understand why there is a range of n values where the number of correct digits of π alternates between two consecutive integers before stabilizing at the larger integer. I suspect this is because Madhava's series is an alternating series, but I would like to look into this further.

As another extension for future work, I would like to compare the Madhava series with other methods of computing digits of π. I would like to find methods that compute digits of π much more quickly than the Madhava series formula.