MATH 317 Assignment 1

Thomas Buffard

Problem 1.

(a). We know that $\sqrt{\pi} = 1.7724$. So we can calculate the error of $\sqrt{\pi}$ with different numbers of known digits:

```
sqrt_err <- function(approx) {
  return(abs(sqrt(approx) - sqrt(pi)))
}

data <- list()
for (n in 0:5) {
  approx <- signif(2 * acos(0.0), (n + 1))
  data[[length(data) + 1]] <- tibble("n" = n, "pi_approx" = approx,
                                     "sqrt_approx" = sqrt(approx),
                                     "err" =   sqrt_err(approx))
}

data <- bind_rows(data)
kable(data)
```

n	pi_approx	sqrt_approx	err
0	3.00000	1.732051	0.0404030
1	3.10000	1.760682	0.0117722
2	3.14000	1.772004	0.0004493
3	3.14200	1.772569	0.0001149
4	3.14160	1.772456	0.0000021
5	3.14159	1.772453	0.0000007

Thus we can see that we need to know π with at least three digits to compute $\sqrt{\pi}$ with four correct decimals, or an error less than 10^{-4}.

(b). To convert to base 10 we do:

$$2^1 + 2^0 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-6} = 2 + 1 + 0.5 + 0.25 + 0.125 + 0.0625 + 0.015625 = 3.953125$$

Problem 2.

Given the function $f(x) = \dfrac{1}{\sqrt{(1+x^2)} - \sqrt{(1-x^2)}}$, we know that $f(x)$ is singular at $x = 0$. So as x gets arbitrarily close to to zero, say $x = 10^{-10} \implies x^2 = 10^{-20}$, the terms $\sqrt{(1+x^2)}$ and $\sqrt{(1-x^2)}$ get closer to 1 and cancel out in the denominator. Thus we need to rewrite the function in order to analyze it as x approaches 0:

$$\begin{aligned}
f(x) =& \frac{1}{\sqrt{(1+x^2)} - \sqrt{(1-x^2)}} \\
=& \frac{1}{\sqrt{(1+x^2)} - \sqrt{(1-x^2)}} \frac{\sqrt{(1+x^2)} + \sqrt{(1-x^2)}}{\sqrt{(1+x^2)} + \sqrt{(1-x^2)}} \\
=& \frac{\sqrt{(1+x^2)} + \sqrt{(1-x^2)}}{(\sqrt{(1+x^2)})^2 - (\sqrt{(1-x^2)})^2} \\
=& \frac{\sqrt{(1+x^2)} + \sqrt{(1-x^2)}}{(1+x^2) - (1-x^2)} \\
f(x) =& \frac{\sqrt{(1+x^2)} + \sqrt{(1-x^2)}}{2x^2}
\end{aligned}$$

Now we can use the rewritten $f(x)$ to accurately analyze the function as it get gets close to 0.

Problem 3.

```
# The function to calculate the root of
f <- function(x) {
  return(sqrt(x) - exp(-x))
}

f_prime <- function(x) {
  return((1 / (2 * sqrt(x))) + exp(-x))
}

# Secant method for calculating roots
# Takes a function f, initial guesses x0, x1,
# n number of iterations, and the root r
sec <- function(f, x0, x1, n = 5, r = 0.4263028) {
  data <- list()
  a <- x0
  b <- x1
  data[[length(data) + 1]] <- tibble("n" = 0, "x_n" = a, "err" = abs(a - r) / abs(r))
  data[[length(data) + 1]] <- tibble("n" = 1, "x_n" = b, "err" = abs(b - r) / abs(r))

  for (i in 2:(n+2)) {
    c <- a - (f(a) * ((a - b) / (f(a) - f(b))))
    b <- a
    a <- c

    data[[length(data) + 1]] <- tibble("n" = i, "x_n" = a, "err" = abs(a - r) / abs(r))
  }

  return (bind_rows(data))
}

# Compute root using Secant method with initial guesses x0 = 1.0 and x1 = 0.75
sec_data <- sec(f, 1.0, .75)
kable(sec_data)
```

n	x_n	err
0	1.0000000	1.3457505
1	0.7500000	0.7593129
2	0.3372934	0.2087938
3	0.4524406	0.0613128
4	0.4276009	0.0030451
5	0.4262843	0.0000433
6	0.4263028	0.0000001
7	0.4263028	0.0000001

```
newton <- function(f, f_prime, x0, n = 5, r = 0.4263028) {
  data <- list()
  a <- x0
  data[[length(data) + 1]] <- tibble("n" = 0, "x_n" = a, "err" = abs(a - r) / abs(r))
```

```r
  for (i in 1:(n+1)) {
    a <- a - (f(a) / f_prime(a))
      data[[length(data) + 1]] <- tibble("n" = i, "x_n" = a, "err" = abs(a - r) / abs(r))
  }

  return (bind_rows(data))
}

# Compute root using Newton-Raphson method with initial guess 1.0
newt_data <- newton(f, f_prime, 1.0)
kable(newt_data)
```

n	x_n	err
0	1.0000000	1.3457505
1	0.2716493	0.3627784
2	0.4116023	0.0344836
3	0.4261836	0.0002795
4	0.4263027	0.0000001
5	0.4263028	0.0000001
6	0.4263028	0.0000001

Problem 4

```
phi_1 <- function(x) {
  return(exp(-2 * x))
}

phi_2 <- function(x) {
  return(-log(x, base = exp(1)) / 2)
}

fixed_point <- function(phi, x0, n = 5, r = 0.4263028) {
  data <- list()
  a <- x0
  data[[length(data) + 1]] <- tibble("n" = 0, "x_n" = a, "err" = abs(a - r) / abs(r))

  for (i in 1:(n+1)) {
    a <- phi(a)
      data[[length(data) + 1]] <- tibble("n" = i, "x_n" = a, "err" = abs(a - r) / abs(r))
  }

  return (bind_rows(data))
}

kable(fixed_point(phi_1, 0.5, n = 20))
```

n	x_n	err
0	0.5000000	0.1728752
1	0.3678794	0.1370466
2	0.4791417	0.1239469
3	0.3835507	0.1002857
4	0.4643571	0.0892659
5	0.3950614	0.0732846
6	0.4537891	0.0644760
7	0.4035002	0.0534891
8	0.4461944	0.0466608
9	0.4096759	0.0390025
10	0.4407172	0.0338126
11	0.4141884	0.0284174
12	0.4367577	0.0245246
13	0.4174814	0.0206929
14	0.4338907	0.0177992
15	0.4198821	0.0150614
16	0.4318123	0.0129240
17	0.4216310	0.0109588
18	0.4303045	0.0093871
19	0.4229044	0.0079718
20	0.4292101	0.0068197
21	0.4238312	0.0057979

```
kable(fixed_point(phi_2, 0.5, n = 15))
```

```
## Warning in phi(a): NaNs produced
```

n	x_n	err
0	0.5000000	0.1728752
1	0.3465736	0.1870248
2	0.5298301	0.2428491
3	0.3175995	0.2549908
4	0.5734821	0.3452459
5	0.2780143	0.3478478
6	0.6400414	0.5013773
7	0.2231112	0.4766367
8	0.7500424	0.7594123
9	0.1438128	0.6626511
10	0.9696215	1.2744902
11	0.0154247	0.9638174
12	2.0858917	3.8929816
13	-0.3675982	1.8622937
14	NaN	NaN
15	NaN	NaN
16	NaN	NaN

For the first fixed-point iteration $\varphi(x) = e^{-2x}$ we can see from the derivative $\varphi'(x) = -2e^{-x}$ that $|\varphi'(x)| = 2e^{-2x} < 1 \ \forall x$. Thus this fixed-point iteration converges by the theorem on convergence. Furthermore, we can see that the data confirms this even though it converges very slowly compared to the two methods in Problem 2.

For the second fixed-point iteration $\varphi(x) = -\frac{ln(x)}{2}$ we can see from the derivative $\varphi'(x) = -\frac{1}{2x}$ that $|\varphi'(x)| = \frac{1}{2x} \geq 1$ for $|x| < \frac{1}{2}$. Thus since our root $|x^*| < \frac{1}{2}$, this fixed-point iteration diverges by the theorem on convergence. This can be seen in our data as our x_n look random and eventually become unrecognizable by the computer (NaN).

Problem 5

This fixed point iteration gives us the function $\varphi(x) = \dfrac{\lambda x + 1 - \sin(x)}{1 + \lambda}$ and the derivative $\varphi'(x) = \dfrac{\lambda - \cos(x)}{\lambda + 1}$. So the fixed point iteration converges for $|\varphi(x)| < 1$.

$$\begin{aligned} |\varphi(x)| < 1 &\implies |\lambda - \cos(x)| < |1 + \lambda| \\ &\implies |\lambda - \cos(x)| \leq |\lambda| + |-\cos(x)| < |1 + \lambda| \leq 1 + |\lambda| \\ &\implies |\lambda| + |-\cos(x)| < 1 + |\lambda| \\ &\implies \cos(x) < 1 \end{aligned}$$

Thus the fixed point interval converges for all $x \in \mathbb{R}$, and furthermore we see that the convergence doesn't depend on λ. To speed up the convergence of x_n to the root x^*, we could choose λ such that $\varphi'(x^*) = 0$. This would cause the convergence to be quadratic or higher instead of linear.

$$\begin{aligned} \varphi'(x^*) = 0 &\implies \frac{\lambda - \cos(x^*)}{\lambda + 1} = 0 \\ &\implies \lambda - \cos(x^*) = 0 \\ &\implies \lambda = \cos(x^*) \end{aligned}$$

Thus by choosing $\lambda := \cos(x^*)$ we can get a convergence that is quadratic or higher.