Math 340 • Fall 2026 • Midterm Exam Practice Problems • Prof. Matveev

1) Find the 2nd degree Taylor polynomial for $f(x) = x^{-3}$ near $x = 1$. Then, use your polynomial to approximate $f(1.1)$, and find the maximal error of this approximation using the Taylor remainder formula.

$$\begin{cases} f(x) = 1/x^3 \\ f'(x) = -3/x^4 \\ f''(x) = 12/x^5 \\ f'''(x) = -60/x^6 \end{cases} \Rightarrow \begin{cases} f(1) = 1 \\ f'(1) = -3 \\ \frac{f''(1)}{2} = 6 \\ \frac{f'''(1)}{3!} = -\frac{1}{10} \end{cases} \Rightarrow \boxed{f(x) = \underbrace{1 - 3(x-1) + 6(x-1)^2}_{P_2(x)} - \underbrace{\frac{10}{c^6}(x-1)^3}_{R_2(x)}} \text{ where } c \in (x, 1) \text{ or } (1, x)$$

$f(1.1) \approx P_2(1.1) = 1 - 0.3 + 0.06 = 0.76$ (compare with exact value: $1/1.1^3 \approx 0.751$: actual error ≈ 0.009)

Error bound: $|R_2(1.1)| = \frac{10 \cdot 0.1^3}{c^6} \le \frac{0.01}{1^6} = 0.01$ ($|R_2|$ is maximized when $c \in (1, 1.1)$ is the smallest, so $c = 1$)

---

2) Consider the function $f(x) = \dfrac{x^2}{x + \ln(\cos(x))}$. Find the first two non-zero terms in the Taylor expansion of $f(x)$ near $x = 0$, using composition of Taylor series for the elementary functions appearing in this formula. Then, use your answer to estimate $f(0.2)$. **Hint:** the exact value (up to 4 digits) is $f(0.2) \approx 0.2224$.

$$f(x) = \frac{x^2}{x + \ln(\cos(x))} = \frac{x^2}{x + \ln\left(1 - \frac{x^2}{2} + O(x^4)\right)} = \frac{x^2}{x - \frac{x^2}{2} + O(x^4)} = \frac{x}{1 - \frac{x}{2} + O(x^3)} = x\left(1 + \frac{x}{2} + O(x^3)\right) = \underbrace{x + \frac{x^2}{2}}_{P_2(x)} + O(x^4)$$

$f(0.2) \approx P_2(0.2) = 0.2 + \frac{0.04}{2} = 0.22$ Correct to two digits!

---

3) Which of the functions below suffer catastrophic loss of accuracy (loss of all digits) when computed in double precision for small $|x|$? Write down the precise condition when this happens. What value will the computation return when this complete loss of significance happens?

(a) $f(x) = -4 + \sqrt{16 + x^4}$, (b) $g(x) = 4 + \sqrt{16 - x^2}$, (c) $h(x) = \dfrac{1}{6x - \sin(6x)}$

Only (a) and (c) involve subtraction of quantities which become equal in the limit $x \to 0$:

(a) $f(x) = -4 + \sqrt{16 + x^4} = -4 + 4\sqrt{1 + \frac{x^4}{16}} \approx -4 + 4\left(1 + \frac{x^4}{32}\right)$ rounded to **zero** if $\frac{x^4}{32} \le 10^{-16} \Rightarrow |x| \le 2^{5/4} \cdot 10^{-4} \approx 10^{-4}$

(c) $h(x) = \dfrac{1}{6x - \sin(6x)}$: will return **Inf** when $\sin(6x)$ is rounded off to $6x$. This happens when the 2nd term in the Taylor expansion of sine is rounded off:

$$\sin(6x) \approx 6x - \frac{(6x)^3}{3!} = 6x\left(1 - \frac{(6x)^2}{3!}\right) \to 6x \text{ when } \frac{(6x)^2}{3!} = 6x^2 \le 10^{-16} \Rightarrow |x| \le \frac{10^{-8}}{\sqrt{6}}$$

4) Convert $x = 134.3$ from decimal to binary system, keeping 8 binary digits after the fractional point. Then, write this binary number is standard scientific notation (i.e. significant part times an integer power of 2).

First, convert the integer part (use division by 2 algorithm, if you prefer): $134 = \underset{2^7}{128} + \underset{2^2}{4} + \underset{2^1}{2} = 10000110_2$

Now, the fractional part, using our simple algorithm:
$$\begin{pmatrix} 0.3 * 2 = 0 + 0.6 \\ 0.6 * 2 = 1 + 0.2 \\ 0.2 * 2 = 0 + 0.4 \\ 0.4 * 2 = 0 + 0.8 \\ 0.8 * 2 = 1 + 0.6 \end{pmatrix}$$

The sequence 1001 has to repeat itself (since we reached 0.6 again), so:

Binary(134.3) = $10000110.\overline{01001}\ldots = 1.\underbrace{00001100}_{\text{First 8 bits}}\overline{1001} \cdot 2^7$

---

5) Consider the root (zero) of function $f(x) = x^3 + x - 1$.

a) Find an obvious bracketing interval $[a, b]$ (with integer *a* and *b*) that contains the positive root of $f(x)$.

[0, 1] is an obvious bracketing interval: $f(0) = -1 < 0,\ f(1) = 2 - 1 = 1 > 0$

b) Perform two iterations of the bisection method, manually.

$$x_1 = \frac{a+b}{2} = \frac{0+1}{2} = \frac{1}{2}: \quad f\left(\frac{1}{2}\right) = \frac{1}{8} + \frac{1}{2} - 1 < 0 \Rightarrow a \doteq \frac{1}{2}$$

$$x_2 = \frac{a+b}{2} = \frac{\frac{1}{2}+1}{2} = \frac{3}{4}: \quad f\left(\frac{3}{4}\right) = \frac{27}{64} + \frac{3}{4} - 1 = \frac{65}{64} - 1 > 0 \Rightarrow b \doteq \frac{3}{4} \Rightarrow \text{root lies within } \left[\frac{1}{2}, \frac{3}{4}\right]$$

c) Write down the expression for the number of bisections of interval [*a*, *b*] needed to guarantee absolute error of 10^{-9} or smaller.

$$\text{We need } \varepsilon_n \le \frac{b-a}{2^n} = \boxed{\frac{1}{2^n} \le 10^{-9}} \Rightarrow \log_2 \Rightarrow -n \le -9\log_2 10 \Rightarrow n \ge \underbrace{9\log_2 10}_{\approx 3.32} \approx \boxed{30}$$

---

6) Consider the iteration $x_{n+1} = g(x_n)$, where $g(x) = \frac{1}{8}(x^2 + 7)$

a) Find all fixed points of this iteration, and determine their stability (**hint:** fixed points are integers)

$$\text{(a) } r = \frac{1}{8}\left(r^2 + 7\right) \Rightarrow r^2 - 8r + 7 = 0 \Rightarrow r_{1,2} = \frac{8 \pm \sqrt{64 - 28}}{2} = \frac{8 \pm 6}{2} \Rightarrow \begin{cases} r_1 = 1: \ |g'(r_1)| = \frac{r}{4} = \frac{1}{4} < 1: \ \text{Stable} \\ r_2 = 7: \ |g'(r_2)| = \frac{r}{4} = \frac{7}{4} > 1: \ \text{Unstable} \end{cases}$$

b) For any **stable** fixed points, find the **number of steps** needed to achieve an absolute error (tolerance) of 2^{-52} (machine precision), assuming initial error of $\epsilon_0 \approx 1/4$. How does this compare with the number of iterations needed to achieve the same accuracy using the bisection method, assuming same initial error?

$$C \equiv \left|g'(r_1)\right| = \frac{1}{4} \Rightarrow \varepsilon_n \approx \varepsilon_0 C^n = \frac{1/4}{4^n} = \frac{1}{4^{n+1}} \le 2^{-52} \Rightarrow 2^{-2n-2} \le 2^{-52} \Rightarrow 2n + 2 \ge 52 \Rightarrow \boxed{n \ge 25}$$

This is **twice as fast** as bisection, since the latter has an error contraction factor of ½ , twice as large as C=¼

7) Suppose the iteration $x_{n+1} = g(x_n)$ has a fixed point r where $g'(r) = 0$ and $g''(r) \neq 0$. Use Taylor series expansion of $g(x)$ about $x = r$ to determine how the error $\epsilon_n \equiv x_n - r$ changes from one step to the next.

Plugging definition of error, $\boldsymbol{x_n = r + \epsilon_n}$, into our iteration $\boldsymbol{x_{n+1} = g(x_n)}$, we obtain:

$$\boldsymbol{x_{n+1} = r + \epsilon_{n+1} = g(r + \epsilon_n).}$$

Now, expand around *r* using the Taylor theorem, to lowest order:

$$\boldsymbol{r + \epsilon_{n+1} = g(r + \epsilon_n) = g(r) + g'(r)\epsilon_n + \frac{g''(\alpha)}{2}\epsilon_n^2} \quad \text{where } \alpha \in (r, x_n)$$

Since by definition $\boldsymbol{r = g(r)}$, we can cancel ***r*** on both sides of this equation, and given $g'(r) = 0$, we obtain:

$$\boldsymbol{\epsilon_{n+1} = \frac{g''(\alpha)}{2}\epsilon_n^2}$$

Sufficiently close to the root $\alpha \approx r$, and we obtain an approximation: $\boldsymbol{\epsilon_{n+1} \approx \frac{g''(r)}{2}\epsilon_n^2}$

---

8) Consider the problem of finding the root (zero) of function $f(x) = x^3 - 2$.

a) Set up the Newton's iteration formula for approximating the root of $f(x)$, and perform two iterations of the Newton's method to compute x_1 and x_2, given an initial guess $x_0 = 1$

$$g(x) = x - \frac{f(x)}{f'(x)} = x - \frac{x^3 - 2}{3x^2} = x - \frac{x}{3} + \frac{2}{3x^2} = \frac{2}{3}\left(x + \frac{1}{x^2}\right)$$

$$x_0 = 1 \;\Rightarrow\; x_1 = \frac{2}{3}\left(1 + \frac{1}{1}\right) = \frac{4}{3} \;\Rightarrow\; x_2 = \frac{2}{3}\left(\frac{4}{3} + \frac{9}{16}\right) = \frac{182}{144} = \frac{91}{72}$$

Side note: this is actually a very good approximation: $\frac{91}{72} \approx 1.264$ vs. $\sqrt[3]{2} \approx 1.260$

b) Estimate the error ϵ_2 of approximant x_2 using $\epsilon_{n+1} \approx M\epsilon_n^2$ where $M = \frac{f''(r)}{2f'(r)}$, assuming that $\epsilon_0 \approx \frac{1}{4}$.

$$M = \frac{f''(r)}{2f'(r)} = \frac{6r}{6r^2} = \frac{1}{r} = \frac{1}{2^{1/3}} \Rightarrow \begin{cases} \varepsilon_0 \approx \frac{1}{4} \;\Rightarrow\; \varepsilon_1 \approx M\varepsilon_0^2 = \frac{M}{16} = \frac{1}{16\sqrt[3]{2}} \\ \varepsilon_2 \approx M\varepsilon_1^2 = M\left(\frac{M}{16}\right)^2 = \frac{M^3}{256} = \frac{1}{512} \end{cases}$$

9) Find function $f(x)$ whose root (solution to $f(r) = 0$) equals $r = 1/\sqrt{3}$. Your function should only involve multiplication, integer powers, and/or division (no fractional powers). Write down the Newton's iteration for this function, and perform two iterations to approximate $r = 1/\sqrt{3}$, starting with $x_0 = 1$.

$f(x) = 3x^2 - 1$ has a root at $r = \frac{1}{\sqrt{3}}$

$$\Rightarrow\; g(x) = x - \frac{f(x)}{f'(x)} = x - \frac{3x^2 - 1}{6x} = x - \frac{x}{2} + \frac{1}{6x} = \frac{1}{2}\left(x + \frac{1}{3x}\right)$$

Two iterations: $x_0 = 1 \;\Rightarrow\; x_1 = \frac{1}{2}\left(1 + \frac{1}{3}\right) = \frac{4}{6} = \frac{2}{3} \;\Rightarrow\; x_2 = \frac{1}{2}\left(\frac{2}{3} + \frac{1}{2}\right) = \frac{7}{12}$

Side note: this is a very good approximation: $\frac{7}{12} \approx 0.583$ vs. $\frac{1}{\sqrt{3}} \approx 0.577$

---

10) Use equation of line through points $(x_n, f(x_n))$ and $(x_{n-1}, f(x_{n-1}))$ to derive the formula for the **secant root-finding method**, which has the form $x_{n+1} = g(x_n, x_{n-1})$. Hint: intersection of this line with the horizontal axis is the next guess x_{n+1}.

Let's start with the point-slope equation of line through the point (x_n, y_n)

$$y - y_n = m(x - x_n)$$

The next approximation x_{n+1} corresponds to the intersection of this line with the horizontal axis $y = 0$, therefore:

$$0 - y_n = m(x_{n+1} - x_n)$$

Divide by the slope of the secant line, which equals $\boldsymbol{m}$, and solve for x_{n+1}: $\boldsymbol{x_{n+1} = x_n - \frac{y_n}{m} = x_n - \frac{f(x_n)}{m}}$:

Finally, plug into this equation the slope of the line connecting $(x_n, f(x_n))$ and $(x_{n-1}, f(x_{n-1}))$: $\boldsymbol{m = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}}$:

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

---

11) The iteration $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f''(x_n)f(x_n)^2}{2[f'(x_n)]^2}$ is an improvement on the Newton's method, achieving higher order convergence. Perform one iteration using this formula with the function $f(x) = x^2 - 2$ and initial guess $x_0 = 1$, as a way to approximate $\sqrt{2}$.

$$\boldsymbol{g(x) = x - \frac{f(x)}{f'(x)} - \frac{f''(x)f(x)^2}{2[f'(x)]^3} = x - \frac{x^2 - 2}{2x} - \frac{2(x^2 - 2)^2}{2(2x)^3} = \frac{x}{2} + \frac{1}{x} - \frac{(x^2 - 2)^2}{8x^3}}$$

$$\boldsymbol{x_1 = g(x_0 = 1) = \frac{1}{2} + 1 - \frac{(1 - 2)^2}{8} = \frac{3}{2} - \frac{1}{8} = \frac{11}{8} = 1.375}$$

Side note: one step gives us 2 digits, and each step triples the number of correct digits: after 2 more steps we get almost machine precision!

12) Perform least-squares fit of the model $\boldsymbol{y_{model}(t;p,q) = p \cdot 2^t + q \cdot t}$, where p and q are unknown parameters, to the data set $\{(t_k, y_k)\} = \{(\mathbf{0},\mathbf{1}), (\mathbf{1},\mathbf{0}), (\mathbf{2},\mathbf{5})\}$. Compute all residuals, and make sure to explain your solution method.

Calculus approach: minimize $L(p,q) \equiv \sum_{k=1}^{N=3} r_k^2 \equiv \sum_{k=1}^{N} \underbrace{\left[y_{\text{model}}(t_k;p,q) - y_{\text{data}}(t_k)\right]}_{\text{"Residuals" (measurement errors)}}^2 = \sum_{k=1}^{N}\left[p\cdot 2^{t_k} + q\cdot t_k - y_k\right]^2$

Let's calculate partial derivatives with respect to parameters, and set them to zero:

$$\begin{cases} \dfrac{\partial L}{\partial p} = \dfrac{\partial}{\partial p}\sum_{k=1}^{N}\left[p\cdot 2^{t_k} + q\cdot t_k - y_k\right]^2 = 2\sum_{k=1}^{N} 2^{t_k}\left[p\cdot 2^{t_k} + q\cdot t_k - y_k\right] = 2\left(p\sum_{k=1}^{N} 4^{t_k} + q\sum_{k=1}^{N} 2^{t_k}t_k - \sum_{k=1}^{N} y_k 2^{t_k}\right) = 0 \\ \dfrac{\partial L}{\partial q} = \dfrac{\partial}{\partial q}\sum_{k=1}^{N}\left[p\cdot 2^{t_k} + q\cdot t_k - y_k\right]^2 = 2\sum_{k=1}^{N} t_k\left[p\cdot 2^{t_k} + q\cdot t_k - y_k\right] = 2\left(p\sum_{k=1}^{N} 2^{t_k}t_k + q\sum_{k=1}^{N} t_k^2 - \sum_{k=1}^{N} y_k t_k\right) = 0 \end{cases}$$

Cancelling factor of 2 and moving the parameter-independent term to the right-hand side, we obtain a linear system:

$$\begin{cases} p\sum_{k=1}^{N} 4^{t_k} + q\sum_{k=1}^{N} 2^{t_k}t_k = \sum_{k=1}^{N} y_k 2^{t_k} \\ p\sum_{k=1}^{N} 2^{t_k}t_k + q\sum_{k=1}^{N} t_k^2 = \sum_{k=1}^{N} y_k t_k \end{cases}$$ which yields the expression above: $$\begin{pmatrix} \sum_{k=1}^{N} 4^{t_k} & \sum_{k=1}^{N} 2^{t_k}t_k \\ \sum_{k=1}^{N} 2^{t_k}t_k & \sum_{k=1}^{N} t_k^2 \end{pmatrix}\begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{N} y_k 2^{t_k} \\ \sum_{k=1}^{N} y_k t_k \end{pmatrix}$$

Plug in data points $\{(t_k, y_k)\} = \{(\mathbf{0},\mathbf{1}), (\mathbf{1},\mathbf{0}), (\mathbf{2},\mathbf{5})\}$.

$$\begin{cases} \sum_{k=1}^{N} 4^{t_k} = 4^0 + 4^1 + 4^2 = 21, \quad \sum_{k=1}^{N} t_k^2 = 1 + 4 = 5 \\ \sum_{k=1}^{N} 2^{t_k}t_k = 1\cdot 2^1 + 2\cdot 2^2 = 10 \\ \sum_{k=1}^{N} 2^{t_k}y_k = 2^0\cdot 1 + 2^1\cdot 0 + 2^2\cdot 5 = 21, \quad \sum_{k=1}^{N} y_k t_k = 10 \end{cases} \Rightarrow \begin{cases} \begin{pmatrix} 21 & 10 \\ 10 & 5 \end{pmatrix}\begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} 21 \\ 10 \end{pmatrix} \Rightarrow \begin{pmatrix} 21 & 10 \\ 2 & 1 \end{pmatrix}\begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} 21 \\ 2 \end{pmatrix} \\ \text{Only couple row operations needed to solve:} \\ p = 1, \ q = 0 \end{cases}$$

Thus, $p = 1, \ q = 0$, and the model is $\boldsymbol{y_{model}(t;p,q) = 2^t}$

Now we can compute the residuals, $r_k = y_{model}(t_k) - y_k = 2^{t_k} - y_k$:

$$r_1 = 1 - 1 = \mathbf{0}; \ \ r_2 = 2 - 0 = \mathbf{2}; \ \ r_3 = 4 - 5 = \mathbf{-1}$$

Optional: geometric approach gives $v_1 = \begin{pmatrix} \phi_1(t_1) \\ \phi_1(t_2) \\ \phi_1(t_3) \end{pmatrix} = \begin{pmatrix} 2^{t_1} \\ 2^{t_2} \\ 2^{t_3} \end{pmatrix}$, $v_2 = \begin{pmatrix} \phi_2(t_1) \\ \phi_2(t_2) \\ \phi_2(t_3) \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$, $\boxed{A = \begin{pmatrix} | & | \\ v_1 & v_2 \\ | & | \end{pmatrix} = \begin{pmatrix} 2^{t_1} & t_1 \\ 2^{t_2} & t_2 \\ 2^{t_3} & t_2 \end{pmatrix}}$

Projecting onto $v_{1,2}$, $\boldsymbol{A^T A x = A^T y}$, gives

$$\begin{cases} A^T A x = \begin{pmatrix} \phi_1(t_1) & \phi_1(t_2) & \phi_1(t_3) \\ \phi_2(t_1) & \phi_2(t_2) & \phi_2(t_3) \end{pmatrix}\begin{pmatrix} \phi_1(t_1) & \phi_2(t_1) \\ \phi_1(t_2) & \phi_2(t_2) \\ \phi_1(t_3) & \phi_2(t_3) \end{pmatrix}\begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{3} 4^{t_k} & \sum_{k=1}^{3} 2^{t_k}t_k \\ \sum_{k=1}^{3} 2^{t_k}t_k & \sum_{k=1}^{3} t_k^2 \end{pmatrix}\begin{pmatrix} p \\ q \end{pmatrix} \\ A^T y = \begin{pmatrix} \phi_1(t_1) & \phi_1(t_2) & \phi_1(t_3) \\ \phi_2(t_1) & \phi_2(t_2) & \phi_2(t_3) \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{3} 2^{t_k}y_k \\ \sum_{k=1}^{3} t_k y_k \end{pmatrix} \end{cases}$$

13) Consider fitting some data set $\{(t_k, y_k)\}$ to the model $\boldsymbol{y_{model}(t; C, p) = C\, t^p \cos(t)}$, where $\boldsymbol{C}$ and $\boldsymbol{p}$ are unknown parameters. Set up the linear algebra system necessary to determine the best values of these parameters, in the least-squares sense.

$$\ln(y_{model}) = \ln C + p \cdot \ln(t) + \ln(\cos t)$$

$$z_{model}(t) \equiv \ln(y_{model}) - \ln(\cos t) = p \cdot \ln t + \ln c$$

Thus, it's just regression with offset $q = \ln C$ $\Rightarrow$ $\begin{pmatrix} \sum_{k=1}^{N} (\ln t_k)^2 & \sum_{k=1}^{N} \ln t_k \\ \sum_{k=1}^{N} \ln t_k & \sum_{k=1}^{N} 1 \end{pmatrix} \begin{pmatrix} p \\ \ln(C) \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{N} \ln t_k \cdot z_k \\ \sum_{k=1}^{N} z_k \end{pmatrix}$ where $z_k \equiv \ln y_k - \ln \cos t_k$

---

14) Use Lagrange polynomial basis to find an interpolating polynomial over points $\{(-1,1),\ (0,0),\ (2,2), (4,6)\}$. You do not need to simplify.

$$f(x) \sim P(x) = y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x) + y_4 L_4(x)$$

$$= \overset{0}{1}\frac{(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)} + 2\frac{(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)} + 6\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)}$$

$$= 1\frac{(x-0)(x-2)(x-4)}{(-1-0)(-1-2)(-1-4)} + 2\frac{(x+1)(x-0)(x-4)}{(2+1)(2-0)(2-4)} + 6\frac{(x+1)(x-0)(x-2)}{(4+1)(4-0)(4-2)}$$

$$= -\frac{1}{15}x(x-2)(x-4) - \frac{1}{6}(x+1)x(x-4) + \frac{3}{20}x(x+1)(x-2).$$

15) The following MATLAB program computes an elementary function using its Taylor series expansion. Which function is it? (hint: follow the value of **F** through a couple iterations of the "while" loop)

```
function F = Taylor(x, eps)
    F = 1;       t = 1;
    x2 = x^2;   k = 1;
    while  abs(t) > eps
          t = – t * x2 / (k * (k + 1));
          F = F + t;
          k = k + 2;
    end
end
```

After 1 iteration $(k=1)$: $t = -\frac{x^2}{1\cdot 2} = -\frac{x^2}{2}$, $\quad F = 1 - \frac{x^2}{2}$

After 2 iterations $(k=3)$: $t = -\frac{x^2}{3\cdot 4}\left(-\frac{x^2}{2}\right) = \frac{x^4}{4!}$, $\quad F = 1 - \frac{x^2}{2} + \frac{x^4}{4!}$

This function calculates $\cos(x)$ using its Taylor series expansion