Worksheet: fast and accurate `exp`

The goal of this exercise is to build a MATLAB function `myexp` which only uses addition, multiplication, and division, *but not the power function* x^y, to compute the exponential function $e^x = \exp(x)$ accurately, say to 15 digits. We do not want to use the power function because x^y is actually evaluated on a computer or calculator by using the more basic exponential and logarithm functions: $x^y = \exp(y \ln x)$.

(a) If a is a positive integer then $\exp(a) = e^a$ can be computed by multiplication alone. This assumes that we have stored way the 15-digit value of $e^1 = \exp(1)$:

```
e1 = 2.71828182845905;
```

If a is *any* integer then $\exp(a) = e^a$ can be computed by at most one division plus additional multiplications. Write a MATLAB code fragment which computes $\exp(a)$ by these means. (*Hint*: Consider $\exp(5) = e^5$ and $\exp(-7) = 1/e^7$, for examples.)

(b) If $|x| > 709$ then $\exp(|x|) > 10^{307}$. Let's say that we just report underflow (i.e. $\exp(x) = 0$) if $x < -709$ and that we report overflow (i.e. $\exp(x) =$Inf) if $x > 709$. Futhermore, assume we can extract the integer and fractional parts of x so that $x = a + b$ where a is an integer and $b \in [0, 1)$; say[1]

```
[a,b] = parts(x);
```

Write a MATLAB code fragment which takes x and uses these ideas to reduce the problem to computing $\exp(b)$ for $b \in [0, 1)$.

[1]In fact `a = floor(x);` and `b = x - a;` is all that is needed to implement `parts`.

(c) Now we need to accurately compute $\exp(b)$ for $b \in [0,1]$. Polynomial interpolation can do this accurately as long as we have some method of evaluating $\exp$ at the interpolation points. Let us suppose that this is possible; for this purpose only we might use the Taylor series of $\exp(x)$. Furthermore, as explained in class and on page 193, we would be wise to use the Chebyshev points $x_0, x_1, \ldots, x_n$ in $[0,1]$ because then

$$\max_{x \in [0,1]} |(x - x_0)(x - x_1) \ldots (x - x_n)| \leq 2^{-2n+1}.$$

(There is an extra power 2^{-n} here because the interval is $[0,1]$ not $[-1,1]$.) Find the degree n so that the error in this interpolation[2] is less than 10^{-15}:

$$\max_{x \in [0,1]} |\exp(x) - p(x)| \leq 10^{-15}.$$

(d) Assume we have an n degree polynomial from the last part,

$$p(x) = c_1 + c_2 x^1 + \cdots + c_{n+1} x^n$$

and that its coefficients are *already* stored in an $n+1$ row vector `c`. Combine the above parts and write a MATLAB code.

```
function z = myexp(x)
% MYEXP  Compute exp(x) to 15 digit accuracy by using only
% elementary operations (+,*,/).  Uses Chebyshev polynomial
% interpolation.  Example to compare:
%   >> myexp(56.7891),  exp(56.7891)
```

[2]Of course, recall $f(x) = p(x) + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0) \ldots (x - x_n)$.