Programming assignment #1

1.1 Error analysis of Horner's rule

1. Need to find a forward error bound of Horner's method and find the relative error

Using (2) from the problem description, it follows that

$$\left|p_n(x)-\hat{c}_0\right| = \left|-\theta_1\alpha_0 - \theta_3\alpha_1 x - \ldots - \theta_{2n-1}\alpha_{n-1}x^{n-1} - \theta_{2n}\alpha_n x^n\right| =$$

$$= \left|\sum_{i=0}^{n-1}\theta_{2i-1}x^i\alpha_i + \theta_{2n}\alpha_n x^n\right| \le \gamma_{2n}\sum_{i=0}^{n}\left|\alpha_i\right|\left|x^i\right| \qquad (1.1)$$

Where $\left|\theta_k\right| \le \gamma_k = \dfrac{ku}{1-ku}$ is used. Next assume that

$$\tilde{p}_n(x) = \sum_{i=0}^{n}\left|\alpha_i\right|x^i \qquad (1.2)$$

Then (1.1a) provides a forward error bound

$$\left|p_n(x)-\hat{c}_0\right| \le \gamma_{2n}\tilde{p}_n(|x|) \qquad (1.3)$$

The relative error bound is

$$\frac{\left|p_n(x)-\hat{c}_0\right|}{\left|p_n(x)\right|} \le \gamma_{2n}\frac{\tilde{p}_n(|x|)}{|p(x)|} \qquad (1.4)$$

2. Equation (2) is a backward error because it shows that the computed value of the polynomial is the exact value at x multiplied by coefficients of the polynomial with a small perturbation. The perturbed parameter is $\hat{\alpha}_i = (1+\theta_i)\alpha_i$, where θ is a relative error. The size of this perturbations is bounded by

$$\left|\theta_k\right| \le \gamma_k = \frac{ku}{1-ku}. \qquad (1.5)$$

3. A perfectly conditioned problem is when $\kappa_{rel} = 1$. The relative condition number is

$$\kappa_{rel} = \frac{\tilde{p}_n(|x|)}{\left|p_n(x)\right|} = \frac{|\alpha_0| + |\alpha_1||x| + \ldots + |\alpha_n|\left|x^n\right|}{\left|\alpha_0 + \alpha_1 x + \ldots + \alpha_n x^n\right|} \qquad (1.6)$$

To make $\kappa_{rel} = 1$, the sign of α_i should match the sign of x^i to produce positive products ($\alpha_i \cdot x^i > 0$). For example, if all coefficients $\alpha_i > 0$ and $x > 0$ $\kappa_{rel} = 1$.

4. The value of $p_n(x)$ can be sensitive with respect to perturbations to the coefficients when x is a close proximity to the roots because the denominator approaches zero as $x \to \rho$.

1.2

Executive summary

Horner's rule was employed to evaluate $p_9(x)$ at 381 points in the interval [1.91, 2.1] in single precision. The result is compared against "exact" solution obtained using the product form in double precision. A priori forward error bounds were analytically estimated. The values of $p_9(x)$ computed with Horner's rule have a notably poor agreement with the exact solution. The relative error at some points is in O(1e8). Nevertheless, computed results stay within the error bound.

1.2 - 1.

Statement of the problem

- Estimate the 9th degree polynomial using Horner's rule and a priori error bounds in single precision.
- Compare results with the exact solution (double precision, using the product form).
- Verify the correctness of the bounding curves.

Description of the algorithms

- Horner's Rule was coded (plnm_horner.F90)

$$p_n(x) = \alpha_0 + x(\alpha_1 + x(\alpha_2 + \ldots x(\alpha_{n-1} + x(\alpha_n))))$$

Calculation is performed in single precision.

- The a priori error bound was evaluated using (1.2, 1.3, 1.5) (plnm_horner.F90) in single and double precisions.

Results

Results of the experiments are presented in Figure1.1. The exact solution is shown separately to see the shape of the polynomial function. The bottom panel demonstrates the exact solution, the estimate derived via Horner's rule, and a priori upper/lower error bounds. Histograms of the absolute errors between the exact solution in double and single precisions and the estimate obtained using Horner's rule are presented in Figure 1.2.

Conclusions

Computed $p_9(x)$ using Horner's Rule in single precision (Ph_sp) has a very poor match with the exact solution (Pex_dp). Absolute errors |Ph_sp – Pex_dp| are in $O(10^{-3})$ (Figure 1.2, top). Nevertheless, the estimates derived via Horner's rule stay within the a prior error bounds (Figure 1.1, bottom). The bound is very broad and not even closely bounds Horner's estimate (Ph_sp). On average, the bound is $O(10^3)$ times bigger than Ph_sp. This is due to the fact that the bound gives the most extreme estimates of the errors. The value of $\tilde{p}_n(|x|)$ is proportional to $(|x|^n)$ (Eq. 1.6), which can be very high for n=9 and |x|>1. Note that the bounds are wider towards the right boundary of the domain, due to larger |x| values.

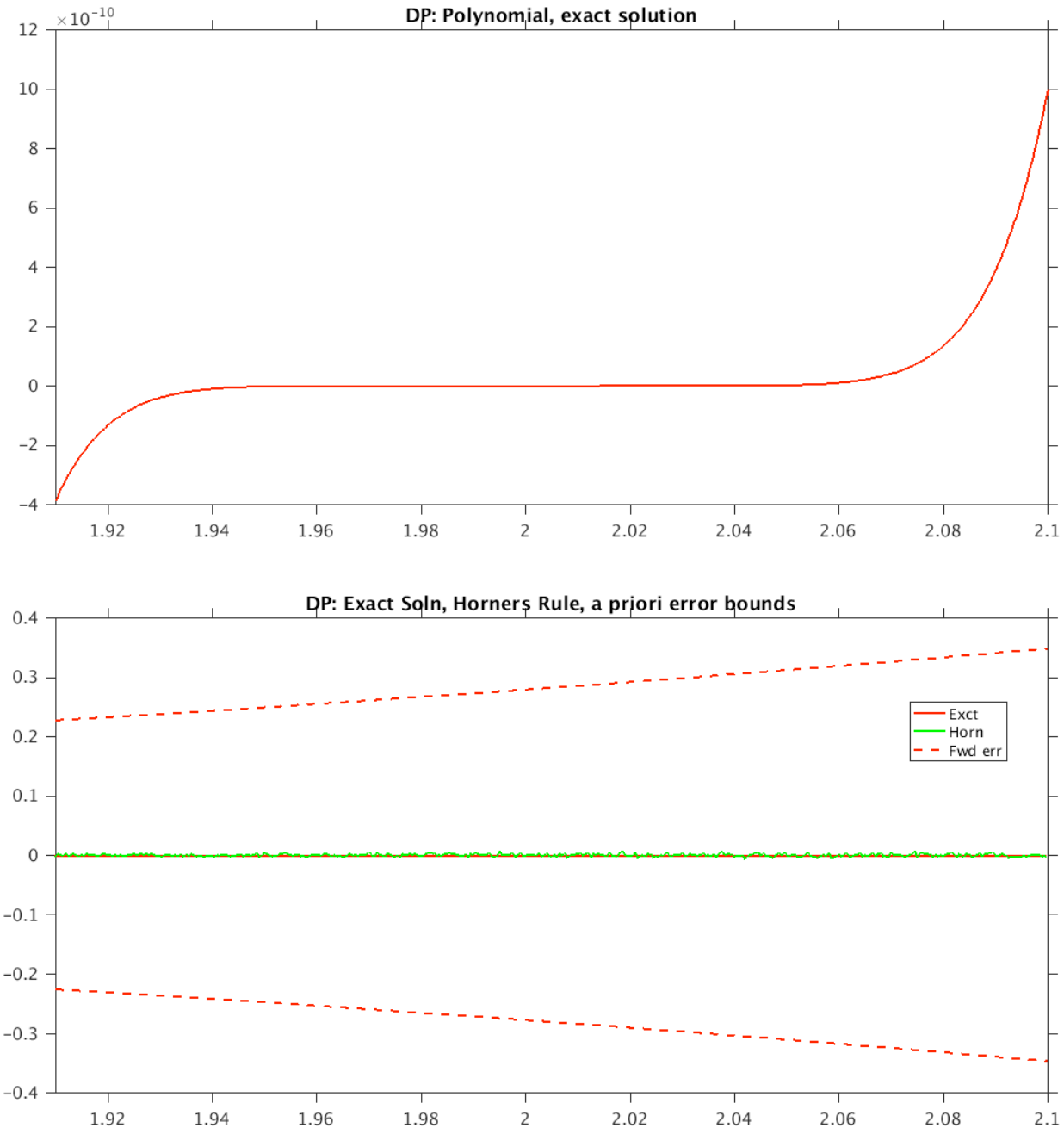

Figure 1.1. Top: *$p_9(x)$ estimated using the product form in double precision (exact solution).* ***Bottom:*** *The exact solution (red), the estimate derived via Horner's rule (green), and a priori upper/lower error bounds.*

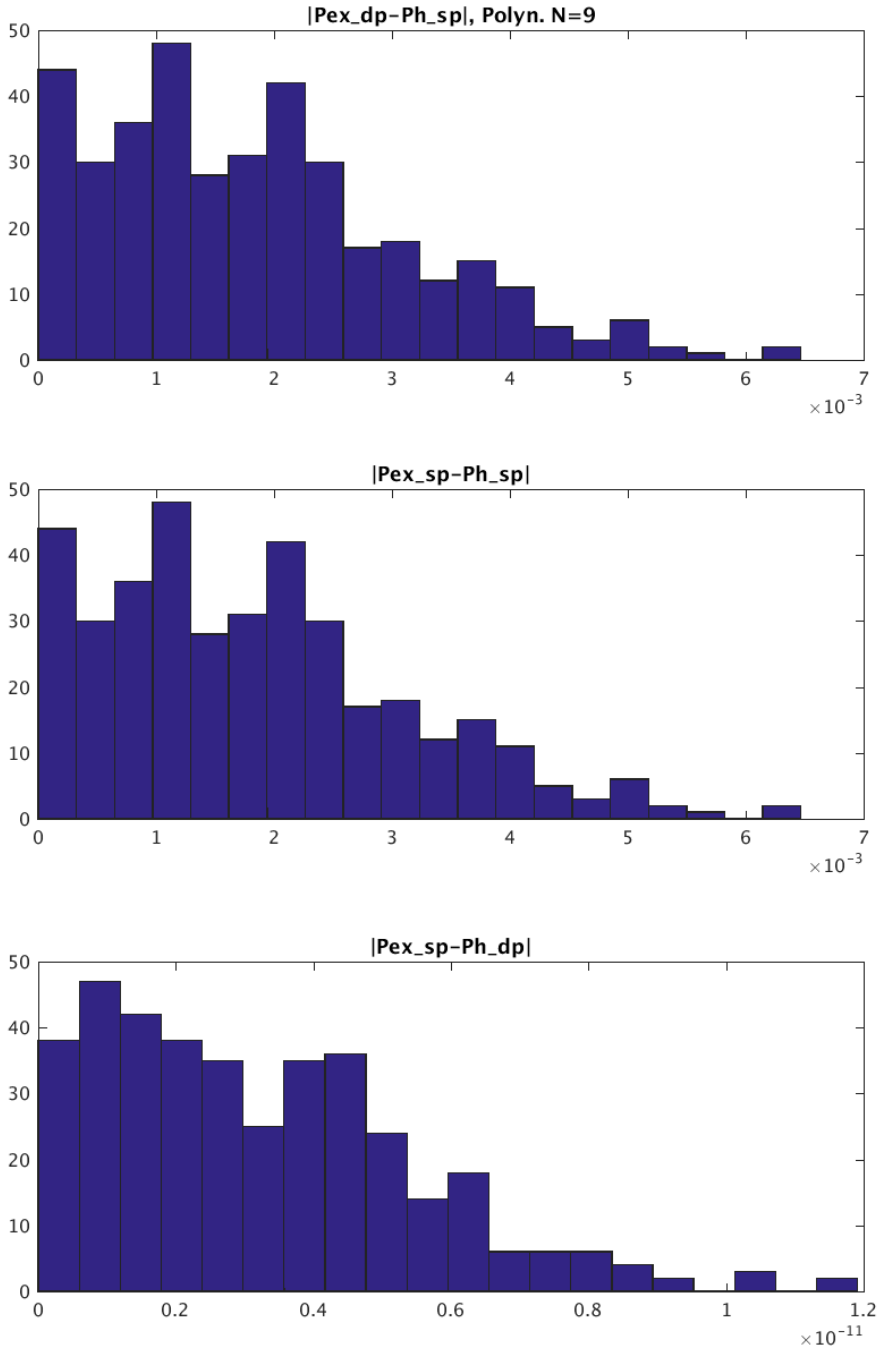

Figure 1.2. *Histograms of the absolute error of the $p_9(x)$ estimates via Horner's rule compared to the exact solution in double precision (top) and in single precision (middle). In the bottom panel, absolute error between the exact solution in double precision and Horner's rule estimate in double precision.*

1.2 - 2.
Statement of the problem

- Repeat the previous procedure with $p_9(x)$ evaluated using the product form in single precision as the exact value.
- Evaluate how this has changed results.

Description of the algorithms

- Exact value of $p_9(x)$ was evaluated using the product form in single precision.
- Same analysis was repeated as in 1.2-1.
- For verification, the estimate of $p_9(x)$ using Horner's rule calculation was completed in double precision (Figure 1.2, bottom).

Results

Results of the experiments are presented in Figure1.3. Absolute errors between the exact solution in single precisions and the estimate obtained using Horner's rule are shown in Figure 1.2 (middle).

Conclusions

Changing precision of the exact solution has not impacted notably the results. The error bounds stay very broad. The magnitude of the absolute error between the Ph_sp and the error bounds has not noticeably changed (Figure 1.3, bottom and Figure 1.2, middle). The reason is that switching from double to single precision has only lightly impacted the accuracy of the floating operation of (x-2) (Figure 1.3, top) because the values of the mesh points are within the single precision accuracy. The absolute error between the exact solutions in single and double precisions is $<10^{-14}$. Note an amplification of the error at the edges of the interval related to Runge's phenomenon.

It is interesting to mention that switching the calculation of the exact solution from double to single precision has not resulted in any obvious reduction of error between the exact solution and Horner's rule (Figure 1.2 middle). However switching to double precision in the Horner's rule calculation significantly reduces the error (by $\sim 10^{-8}$ times) (Figure 1.2 bottom). This demonstrates that the computational error accumulates and get amplified (by the factor of n) in the Horner's rule.

1.2 -3.

The fact that the a priori error bound is evaluated in single precision is significant because $\left|p_n(x) - \hat{c}_0\right| \le \gamma_{2n}\tilde{p}_n\left(|x|\right)$

And $\gamma 2n \sim O(u)$ (u is the unit roundoff error of the floating point system), which is $O(10^{-8})$ for single precision and $O(10^{-16})$ for double precision. The error bounds would be much tighter (by $\sim 10^8$ times) if double precision is used. It would be better to use a double precision for the a priori bound evaluation.

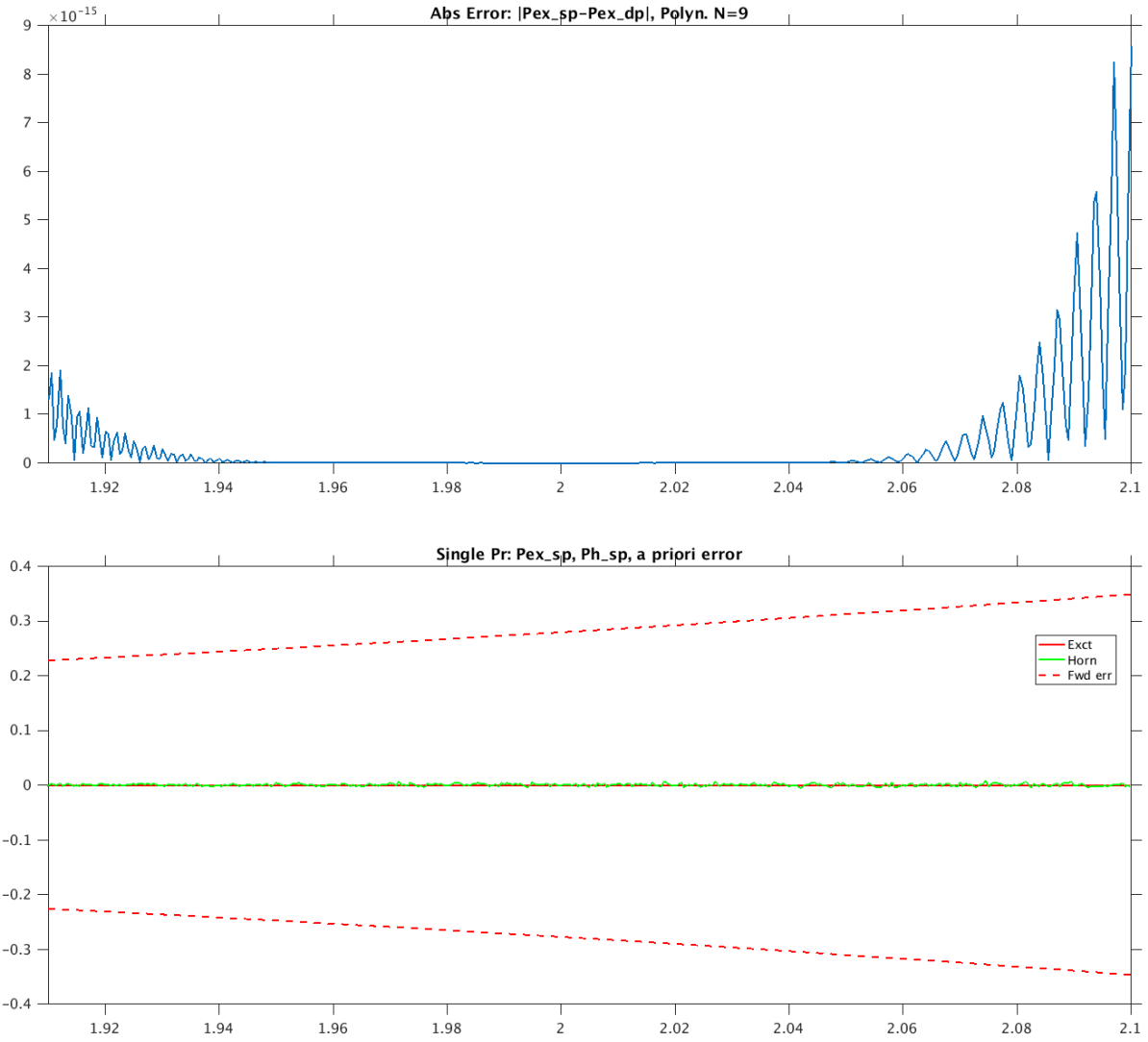

Figure 1.3. *Absolute error of the exact solution for $p_9(x)$ estimates in double precision and single precision (top). Note an amplification of the error at the edges of the interval (Runge's phenomenon). In the bottom panel, the absolute error between the exact solution in single precision and Horner's rule estimate (in single precision) is shown.*

1.3

Derive a running error bound formula. The method provides a posteriori error bound.

The computed value of Horner's rule is

$$(1+\varepsilon_i)\hat{c}_i = x\hat{c}_{i+1}(1+\delta_i)+\alpha_i, \qquad (3.1)$$

where c_i is the exact value of the polynomial in Horner's rule evaluated in exact arithmetic. $|\delta_i| \le u, \quad |\varepsilon_i| \le u$, where u is the unit roundoff of the floating point system.

Define $\hat{c}_i = c_i + e_i$, with $e_n = 0$ (3.2)

From (3.1) and using the definition (3.2)

$$c_i + e_i + \varepsilon_i \hat{c}_i = x\hat{c}_{i+1} + \delta_i \hat{c}_{i+1} x + \alpha_i = x\left(c_{i+1} + e_{i+1}\right) + \delta_i \hat{c}_{i+1} x + \alpha_i \quad (3.3)$$

After rearranging terms in (3.3) and noticing that according to Horner's rule $c_i = xc_{i+1} + \alpha_1$, we get

$$e_i = xe_{i+1} + \delta_i \hat{c}_{i+1} x - \varepsilon_i \hat{c}_i. \quad (3.4)$$

From Cauchy-Schwartz inequality

$$|e_i| \le |x||e_{i+1}| + |\delta_i||\hat{c}_{i+1}||x| + |\varepsilon_i||\hat{c}_i|. \quad (3.5)$$

Since $|\delta_i| \le u, \quad |\varepsilon_i| \le u$, (3.5) becomes

$$|e_i| \le |x||e_{i+1}| + |u|\left(|\hat{c}_{i+1}||x| + |\hat{c}_i|\right). \quad (3.6)$$

$$|e_i| \le u\beta_i, \quad \beta_n = 0$$

Where $\beta_i = |x|\beta_{i+1} + |\hat{c}_{i+1}||x| + |\hat{c}_i|$. (3.7).

1.3-1 and 1.3-2

Statement of the problem

- Compute a posteriori error bound estimate.
- Compare results with the a priori bound prediction obtained in 1.2 when Horner's rule and the error bound are obtained in single precision and the exact is evaluated in double precision (Pex_dp).

Description of the algorithms

- The a posteriori (running) error bound was evaluated using (3.6, 3.7).
- The code was added to the Horner's rule loop (plnm_horner.F90) in single precision.

Results

A priori error bounds, a posteriori (running) error bounds and the absolute difference |Pex_dp-Ph_sp| are plotted in Figure 3.1.

Conclusions
The running error bound is much tighter (about 18 times) and gives a more accurate estimate of the error than the a priori error bound estimate (Fig. 1.4). The running error bound is only ~2.5 to 20 times bigger than the actual absolute error |Pex_dp-Ph_sp|. The absolute error stays within the running error bound.

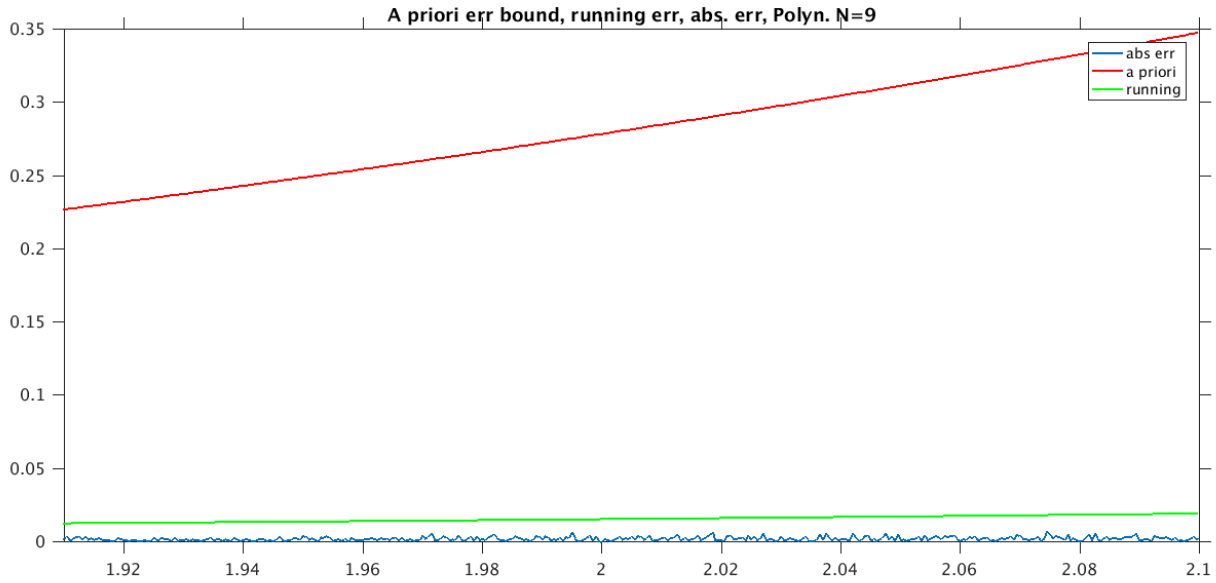

Figure 1.4. *A priori error bound (red), a posteriori (running) error bound (green), and the absolute error of the Horner's rule estimate vs exact solutions for $p_9(x)$ (blue).*

1.3-3.

Statement of the problem

- Repeat the procedure and analysis for two other polynomials

Description of the algorithms

- Two polynomials are proposed of order 3 and 4
- The intervals are chosen within the proximity of the roots
- Polynomial #2:
  $P_3(x) = (x-3)^3 = x^3-9x^2+27x-27$
  Considered interval is [2.9, 3.1].
- Polynomial #4:
  $P_4(x) = (x-4)^4 = x^4-16x^3+96x^2-256x+256$
  Considered interval is [3.9, 4.1].

Results
Results for two other polynomials are presented in Figures 1.5 ($p_3(x)$) and 1.6 ($p_4(x)$).

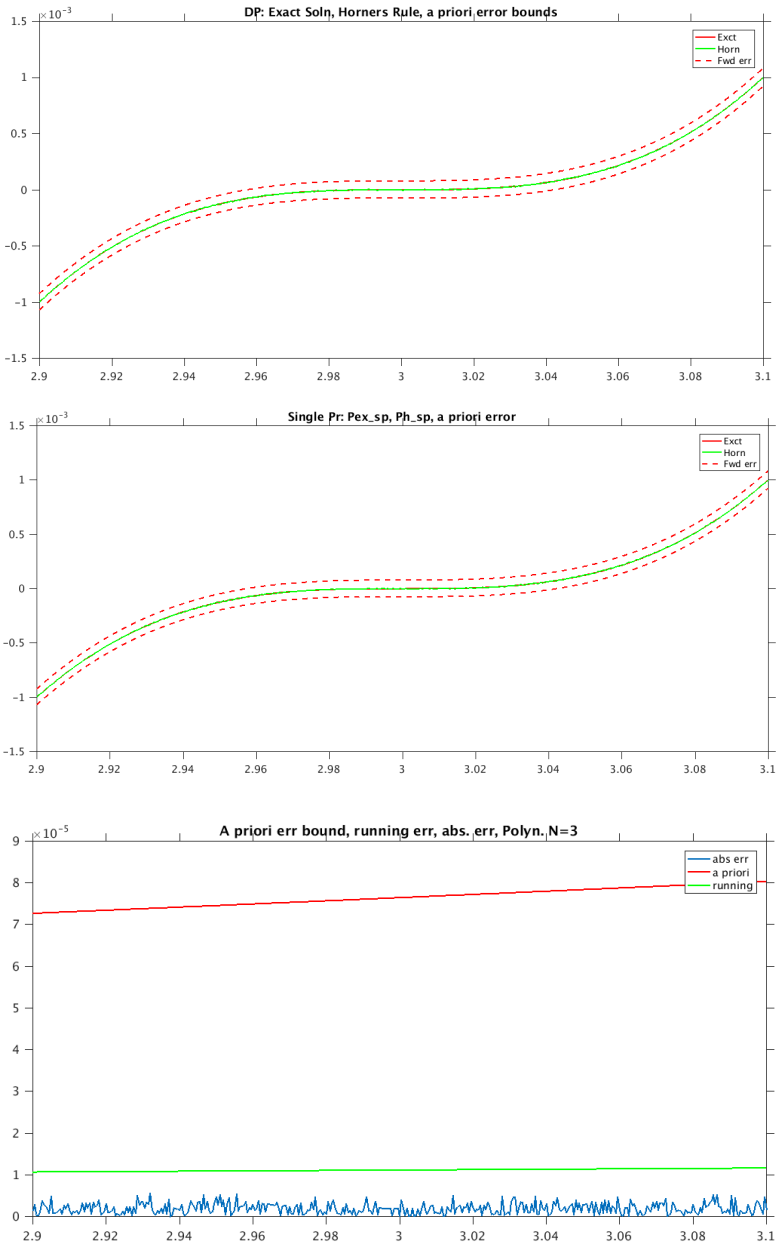

Figure 1.5. *Experiments with $p_3(x)$. Top: $p_3(x)$ exact solution evaluated in double precision. The exact solution (red), the estimate derived via Horner's rule (green), and a priori upper/lower error bounds (red dashed). Middle: same as the top panel but the exact solution evaluated in single precision. Bottom: A priori error bound (red), a posteriori (running) error bound (green), and the absolute error of the Horner's rule estimate vs exact solutions for $p_3(x)$ (blue).*

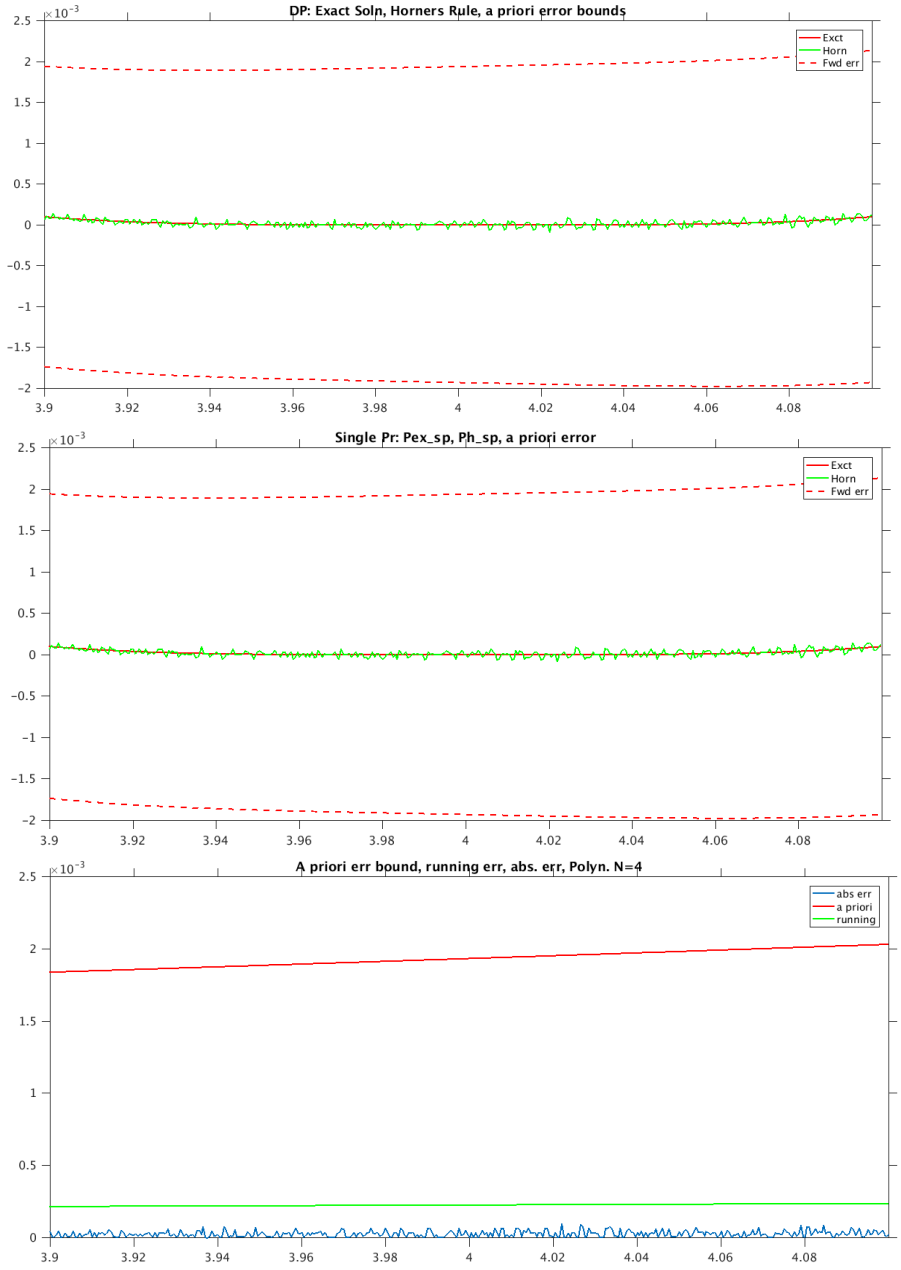

Figure 1.6. *Experiments with $p_4(x)$. Top: $p_4(x)$ exact solution evaluated in double precision. The exact solution (red), the estimate derived via Horner's rule (green), and a priori upper/lower error bounds (red dashed). Middle: same as the top panel but the exact solution evaluated in single precision. Bottom: A priori error bound (red), a posteriori (running) error bound (green), and the absolute error of the Horner's rule estimate vs exact solutions for $p_4(x)$ (blue).*

Conclusions

Experiments with the two proposed polynomials demonstrate same results, in general. That is:

- The a priori error bounds are too broad and provide a very rough error bound estimates (Figs. 1.5 top, 1.6 top).
- Switching to a single precision in pn(x) evaluation does not impact the results
- The running error provides ~O(10) times narrower bound than the a priori estimate.

Nevertheless, there are substantial differences in the experiments with the lower order polynomials.

(1) The Horner's rule evaluated in single precision is closer to the exact solution. The absolute error is $O(10^{-6})$ for p3(x) and $O(10^{-4})$ for p4(x). This demonstrates the tendency of the Horner's rule to have higher error for higher degree polynomials.

(2) The error bounds are tighter in agreement with Eq. (1.2) and (1.3).

Instructions for running the code and plotting the results

The tar bundle program1_Dukhovskoy.tar includes Fortran codes where all calculations are conducted. It also includes Makefile for compiling and matlab codes for plotting results.

1) Untar file
2) Change library directory pathnames in the Makefile
3) Compile the codes,

```
ddmitry@mars: /FCM1> make
/opt/intel/12/bin/ifort -c -convert big_endian -O3 -I/opt/hpc/intel13/include
utils.F90
/opt/intel/12/bin/ifort -c -convert big_endian -O3 -I/opt/hpc/intel13/include
plnm_horner.F90
/opt/intel/12/bin/ifort utils.o plnm_horner.o  -L/opt/hpc/intel13/lib64 -limf -lm -o
hrn.x
```

it should produce an executable hrn.x

```
ddmitry@mars: /FCM1> ls -rlt
total 976
-rwxr-xr-x 1 ddmitry coaps    682 Oct  5 14:06 Makefile
-rw-r--r-- 1 ddmitry coaps    482 Oct  6 00:57 PARAM2.dat
-rw-r--r-- 1 ddmitry coaps    421 Oct  6 00:58 PARAM1.dat
lrwxrwxrwx 1 ddmitry coaps     10 Oct  6 08:42 PARAM.dat -> PARAM3.dat
-rw-r--r-- 1 ddmitry coaps   4458 Oct  6 11:45 utils.F90
-rw-r--r-- 1 ddmitry coaps   2557 Oct  6 11:47 plnm_horner.F90
-rw-r--r-- 1 ddmitry coaps   4864 Oct  6 12:11 plot_polynom_fort.m
-rw-r--r-- 1 ddmitry coaps    501 Oct  6 13:58 PARAM3.dat
drwxr-xr-x 2 ddmitry coaps   4096 Oct  6 14:32 bkp
-rw-r--r-- 1 ddmitry coaps  20480 Oct  6 14:33 program1_Dukhovskoy.tar
-rw-r--r-- 1 ddmitry coaps  26888 Oct  6 14:36 utils.o
-rw-r--r-- 1 ddmitry coaps   5886 Oct  6 14:36 utils.mod
-rw-r--r-- 1 ddmitry coaps  18608 Oct  6 14:36 plnm_horner.o
-rwxr-xr-x 1 ddmitry coaps 874548 Oct  6 14:36 hrn.x
```

Edit PARAM?.dat files – change output directory name

PARAM?.dat files include all necessary information for the experiments
PARAM1.dat – polynomial p9(x)
PARAM2.dat polynomial p3(x)
PARAM3.dat polynomial p4(x)

The code can be run two ways:
hrn.x PARAM1.dat

Or make a soft link to the PARAM?.dat that needs to be run
And type
hrn.x

Plot output:
In plot_polynom_fort.m
Change plnm = 1 (2 or 3) corresponding to PARAM1(2 or 3).dat
Change output directory name
Run the code in Matlab.