Below is a list of problems for our lab session

Wednesday, December 3rd, (kl. 13.00-16.00), Room MA436-446.

The documentation for \texttt{cnf.m}, \texttt{evf.m} has been completed with minimal working examples. A typographical error in \texttt{rk.m} has been fixed and a few new subroutines have been added. Download a new copy of the site backup.
Problem 1 Polynomial interpolation and Newton’s method. Newton’s method converges quadratically, provided the initial guess is sufficiently close to the root. We can use interpolation to determine a good initial guess.

1. Use your targeting routine to construct the firing solution for targets located at distance \( r_j = 1000 + 5000j \), \( j = 0, 1, 2, 3 \). As usual, we are using the gun defined by the minimal working example of range_rkx2.

2. Construct the Newton form of the polynomial \( p \) which satisfies

\[
p\left( \frac{r_j}{1000} \right) = \text{the elevation for the low trajectory to } r_j
\]

for \( j = 0, 1, 2, 3 \). Assuming that \text{table}(2,:) does contain the low angles, then

\[
\text{nodes} = 1:5:17
\]

\[
[c, \text{flag}, \text{condnum}] = \text{cnf}(\text{nodes}, \text{table}(2,:));
\]

would compute the coefficients that you need

\textbf{Remark 1} The scaling \( r_j/1000 \) seems very arbitrary, but it is necessary because it controls the condition number of the linear system which we must solve in order to compute the coefficients which determine \( p \). We will discuss condition numbers for linear systems at a later time, but as an experiment you should try to omit the scaling and see the condition number \text{condnum} explode.

3. Plot the graph of \( p \) on the interval \( r \in [0,17000] \). Remember the scaling. Reasonable commands are:

\[
\text{t=linspace}(0,17000,1000);
\]

\[
\text{p}=\text{enf}(c,\text{nodes},t);
\]

\[
\text{plot}(t,p(c,\text{nodes},t/1000),\text{nodes},\text{table}(2,:),\text{"*"});
\]

4. How many iterations does my_target_1d require to compute the low angle to a target at \( r = 13.5 \) km if you use an initial guess of \( \theta_0 = 0 \) and a tolerance of 30 meters.

5. How many iterations does my_target_1d require if you use \( \theta_0 = p(13.5) \) as an initial guess?

6. Can you find a polynomial which can compute firing solutions to new target in the range from 1 km to 17 km \textit{without} computing any new trajectories?

\textbf{Hint:} Perhaps a moderate increase in the number of interpolation nodes might be prudent? You need to verify that your polynomial actually
works. Brute force verification is pretty fast in this case, but what is the minimal number of tests that you have to perform assuming a kill radius of $\rho = 30$ meters?
Problem 2 (Richardson extrapolation for numerical differentiation)
The class website now contains a function \texttt{rdif} which does Richardson extrapolation and error estimation.

1. Read through the documentation. How many function handlers does this function require? Take note of the fact that it is possible to pass a variable number of arguments to a \texttt{MATLAB} function.

2. Execute the minimal working example and study the output carefully. In particular:
   
   (a) Verify that Richardson’s fractions converge monotonically to $4 = 2^p$, where $p = 2$ is the order of the dominant error term for the operator $D2$ used by the minimal working example.

   (b) Verify that the equality of the error estimate begins to decay within one unit of the value of $k$ for which the computed value of fractions start to deviate from the monotone convergence which would occur had all computations been done in exact arithmetic.

3. Pick another value of $x$ and run the \texttt{rdif} again and examine the output as before correlating the behavior of the fractions with the quality of the error estimate.

4. Replace $D2$ with $D1$ which is also defined in the minimal working example and run \texttt{rdif} again. Remember to redefine $p$. Examine the output as carefully as before.

5. Find the highest value of $k$ for which the fractions are still behaving as they would if they had been computed in exact arithmetic. Let $Dh$ be the corresponding approximation for $f'(x)$. Compute $Dh + (Dh - D2h)/(2^{(p-1)})$ and compare it with $f'(x)$. 


Problem 3 (Richardson extrapolation for numerical differentiation)
You will now play a game of wits with your neighbor.

1. Your neighbor will pick a nontrivial but differentiable function $f$ and pro-
gram both $f$ and $f'$ into MATLAB without revealing the definition of the
function to you. Your neighbor will also define a discrete operator, either
$D = D_0$ or $D = D_1$ where

$$D_1(f, x, h) = \frac{f(x + h) - f(x)}{h}, \quad \text{(2)}$$

$$D_2(f, x, h) = \frac{f(x + h) - f(x - h)}{2h} \quad \text{(3)}$$

is one of the two standard ways of approximating $f'(x)$, but you will not
be informed of the choice until later. Finally, you will receive values $x$ and
$h$ so that the interval $[x - h, x + h]$ is well inside the domain of $f$.

2. Apply Richardson’s technique as implemented in `rdif.m` to the problem of
approximating $f'$ using the unknown function $f$ and the unknown operator
$D$. You are not allowed to pass the exact value of $f'(x)$ to `rdif.m`. Since
you do not know which of the two difference operators you were given,
you will play it safe and pass $p=1$ to the subroutine.

3. Study the fractions carefully! Which of the two operators $D_1$ or $D_2$ did
your neighbor really give you?

4. What is the range of integers $k$ for which the corresponding fractions
converge monotonically to a power of two?

5. Compute the derivative as accurately as you can from the output of `rdif`
and compare with the exact result which your neighbor will now allow you
to compute. **WARNING:** You will remember to pass the correct value
of $p$ to the function `rdif.m`
Problem 4 Richardson extrapolation for $f''$ Let $f : I \to \mathbb{R}$ be a function which is four times differentiable with a continuous fourth order derivative.

1. Let $x \in I$ and let $h > 0$ be such that $[x - h, x + h] \subseteq I$. Then by Taylor’s formula, there exists $\xi$ and $\nu$ in the interval $(x - h, x + h)$ such that

\[
f(x + h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \frac{1}{6}f'''(x)h^3 + \frac{1}{24}f^{(4)}(\xi)h^4 \tag{4}\]

\[
f(x - h) = f(x) - f'(x)h + \frac{1}{2}f''(x)h^2 - \frac{1}{6}f'''(x)h^3 + \frac{1}{24}f^{(4)}(\nu)h^4. \tag{5}\]

Show that

\[
f''(x) = \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} - \frac{1}{24}\left(f^{(4)}(\xi) + f^{(4)}(\nu)\right)h^2. \tag{6}\]

2. Show that there exists a $\theta$ between $\xi$ and $\nu$ such that

\[
\frac{f^{(4)}(\xi) + f^{(4)}(\nu)}{2} = f^{(4)}(\theta). \tag{7}\]

**Hint:** Let $g(x) = f^{(4)}(x)$. Then $g$ is continuous on the closed and bounded interval $[x - h, x + h]$. Therefore there are constants $m$ and $M$, such that

\[
\forall t \in [x - h, x + h] : m \leq g(t) \leq M. \tag{8}\]

Show that if $t_i \in [x - h, x + h]$ for $i = 1, 2$, then

\[
m \leq \frac{g(t_1) + g(t_2)}{2} \leq M. \tag{9}\]

Use the continuity of $g$ a second time to find $\tau$ such that

\[
\frac{g(t_1) + g(t_2)}{2} = g(\tau). \tag{10}\]

3. Implement a difference operator $D22$ in MATLAB with the command

\[
D22=@(f,x,h)(f(x+h)-2*f(x)+f(x-h))/(h*h)\]

4. Pick a function which is many times differentiable, implement it in MATLAB together with its second derivative and pass it through rdiff studying the input as before.

5. Why exactly are we not surprised to see the fractions converging monotonically to 4 until catastrophic cancellation starts to set in?
Remark 2 Take a moment to consider what we have done so far with respect to the artillery problem. Four weeks ago, we constructed an elementary firing table giving the range as a function of the elevation. From the table we could read off a bracket which was systematically refined using the bisection algorithm until the residual was less than the kill radius. Realizing that the bisection algorithm could be very slow, particularly if pinpoint accuracy is required, you implemented Newton’s method. Artillery range functions $\theta \rightarrow r(\theta)$ are typically convex and in this case Newton’s method does in fact converge for any initial guess, but we can always accelerate the process by picking a good initial guess. Today you have a method of constructing an initial guess which is so good that no further refinement is necessary!

However, our gun is firing high explosive shells with a large kill radius. Pinpoint accuracy is require for anti-aircraft fire and anti missile defense. A typical close-in weapons system such as the Phalanx fires 75 rounds/second with an effective firing range of 3.6 km. A typical sea-skimming missile such as the Sea Sparrow does more than 1.1 km/s. The missile must be destroyed at least 500 m out or the fragments from the explosion can still kill personnel or damage antenna. This leaves precious little time to obtain and hold a target lock and we need the speed of Newton’s method to complete the calculation fast enough.

Our shell model is perfectly adequate for short range guns and spin stabilized mortar shells. More complicated models are needed in general and Newton’s method coupled with precomputed firing tables for standard conditions remains the only viable option.