18.03 Problem Set 4

Due by 1:00 P.M., Friday, March 12, 1999, in the boxes at 2-106, next to the Undergraduate Mathematics Office.

Syllabus

14. (F 5 Mar) Operators; higher order equations: Notes O 4–8.

15. (M 8 Mar) Inhomogeneous equations: undetermined coefficients: 2.5, handout.

16. (W 10 Mar) Undetermined coefficients: Exponential Shift Law.

17. (F 12 Mar) Nonconstant coefficients, reduction of order, nonlinear examples: EP 2.6, handout on the Airy equation.

18. (M 15 Mar) Review.

19. (W 17 Mar) Hour Exam II.

Part I.

14. (F 5 Mar) **(a)** Compute D^3x^3; **(b)** Compute $(D^2+I)\sin(x)$; **(c)** Verify the ESL for $(D^2+2D+17I)(e^{2x}x)$; **(d)** Recall the notation M_f for the *multiplication* operator defined by $M_f u = fu$ where f and u are functions. Calculate that $M_f(u+v) = M_f u + M_f v$ and $M_f(cu) = cM_f u$ for c constant: M_f is *linear*. Explain why $M_{f+g} = M_f + M_g$ and $M_{cf} = cM_f$, and also why $DM_f - M_f D = M_{f'}$ (where $f' = Df$ is the derivative of f). A first order linear differential operator has the form $D + M_p D$ (where p is a function, not necessarily constant). Calculate that

$$(D+M_p)(D+M_q) = D^2 + M_{p+q}D + M_{q'+pq}.$$

15. (M 8 Mar) EP 2.5: 1, 7, 11, 21, 23, 25.

16. (W 10 Mar) Nothing new.

Part II.

14. (F 5 Mar) In **12.** you found a way to use one solution of a second order homogeneous linear equation to "factor" it into a sequence of two first order linear equations. This problem gives another such factorization, and shows clearly where the two constants of integration come from.

We'll solve $y'' - 3y' + 2y = x^3$ using this technique. First, write down the operator L so that the left hand side is Ly. Then factor L into a product $L = (D - r_1 I)(D - r_2 I)$. ($r_1$ and r_2 are the roots of the characteristic polynomial.) Our equation can then be decomposed into two:

$$\begin{aligned} (D - r_1 I)u &= x^3 \\ (D - r_2 I)y &= u \end{aligned}$$

Use integrating factors to solve these two first order linear ODEs: do the first one first, and substitute its general solution into the second one.

15. (M 8 Mar) Consider the inhomogeneous undamped second order linear equation

$$\ddot{x} + \omega_0^2 x = a\cos(\omega t). \tag{1}$$

First assume that $\omega \neq \omega_0$ (and both are positive).

(a) By undetermined coefficients, find the general solution. Why did I write ω_0^2 for the coefficient of x?

(b) Find the particular solution $x(t)$ for which $x(0) = \dot{x}(0) = 0$.

(c) Using the formula for the cosine of $\alpha + \beta$, show that

$$\cos(\alpha - \beta) - \cos(\alpha + \beta) = 2\sin\alpha\sin\beta.$$

Use this to re-express your answer to **(b)** as

$$x = \frac{2a}{\omega_0^2 - \omega^2}\left(\sin\frac{(\omega_0 - \omega)t}{2}\right)\left(\sin\frac{(\omega_0 + \omega)t}{2}\right).$$

This expression explains reveals that "beats" occur when the a system with "natural frequency" ω_0 is driven by a periodic force with frequence ω near to ω_0: Then half the difference is small and corresponds to a low-frequency amplitude oscillation. If this were a piano string responding to another one, you would hear a tone of the average of the two pitches, $(\omega_0 + \omega)/2$, with rythmically pulsing amplitude. The amplitude of the beat grows as ω gets closer to ω_0.

(d) Now assume that $\omega = \omega_0$, and find the particular solution $x(t)$ for which $x(0) = \dot{x}(0) = 0$. Using the method of undetermined coefficients and complex exponentials, find the general solution in this case. Sketch the graph of this solution in case $a = \omega = 1$. Why does this show how lucky we are that the world in fact obeys nonlinear equations?

16. (W 10 Mar) This problem will introduce you to MATLAB's symbolic package, which often allows one to solve differential equations "analytically." So settle down in front of MATLAB and read on. Several parts to be handed in are sprinkled through this text. This will also serve as a review of some techniques from earlier in the course.

The first step is to understand how to represent symbolic expressions in MATLAB. You did this in **2.**, when you entered `'x^3-x-1=0'`. The expression inside the quotes may be stored by typing `eq='x^3-x-1=0'`. Do this, and then type `eq`. The expression `x^3-x-1=0` should appear; note that it appears without quotes. Now type `solve(eq)`; you should get the same list of symbolic expressions you got by typing `solve('x^3-x-1=0')`.

You can represent *differential* equations in MATLAB by similar means. Type `de='Dx+k*x=a*k*cos(w*t)'`, and then try typing `de`. The effect of these commands is to assign to a string of symbols (such as `Dx+k*x=a*k*cos(w*t)`) a name (such as `de`). MATLAB doesn't *know* that this string of symbols represents a differential equation.

But the symbolic differential equation solver in MATLAB knows how to interpret such strings of symbols. To see this, type `dsolve(de)` and wait for a response. Note `dsolve`'s choice of name for the constant of integration: `C1`. Take pencil to paper and translate what MATLAB writes out into human mathematical notation. You will become aware of a peculiarity of MATLAB algebraic convention, which is illustrated for you also by typing

`2/1*3` at it. To my eyes this is an ambiguous expression, meaning either $(2/1)\cdot 3 = 6$ or $2/(1\cdot 3) = 2/3$. Which does MATLAB intend?

Anyway, you can see that by itself MATLAB isn't too smart; the combination of human and machine is smarter. You should end up with an expression equivalent to

$$a\,\frac{\cos(\omega t) + (\omega/k)\sin(\omega t)}{1 + (\omega/k)^2} + ce^{-kt}.$$

This differs from the expression

$$\frac{a}{\sqrt{1 + (\omega/k)^2}}\cos(\omega t - \varphi) + ce^{-kt}, \qquad \tan(\varphi) = \omega/k,$$

which I came up with in class. It's a little exercise in trigonometry to see that they are in fact equal, though; and mine carries a lot more information. You can try typing `pretty(ans)` (immediately after the `dsolve(de)` command, but as usual it won't help much. Another option which is sometimes useful is `simple(ans)`. It tries various trig identities and so on; but it doesn't help much in this case. Try it.

Now let's use `dsolve` to check your answer to **4.** So type `dsolve('Dy=y^2-x^2-1')`. You'll get some messy thing starting `RootOf...`, bearing no relation to your answer. The reason is that `dsolve` assumes as a default that the independent variable is t, even if t never shows up in the expression. It assumed that x was some constant. Even so, it did a poor job of solving. You might write $a^2 = x^2 + 1$, if you want to think of x as a constant; so try `dsolve('Dy=y^2-a^2')`. This is an autonomous equation, right? **(a)** Solve it yourself (using partial fractions to do the integration) and compare your answer with what `dsolve` came up with.

Now let's return to equation $dy/dx = y^2 - x^2 - 1$. You can force `dsolve` to use x as independent variable by appending `'x'` to the list of arguments. So type `y=dsolve('Dy=y^2-x^2-1','x')` and see what happens. As before, **(b)** rewrite this in human form and see if it agrees with what you got (or what I got—solutions are on the webpage) for **4.** You'll have to do some rewriting, and as usual `simple` doesn't do this for you.

Writing `y=...` defined `y` to be a symbolic representation of the solution to the ODE. Check this by typing `y`. Now suppose we have the initial condition $(x, y) = (0, 1)$ in mind. This will specify a value for the constant of integration `C1`. `dsolve` can do this for you: type `y=dsolve('Dy=y^2-x^2-1','y(0)=1','x')` and look at the result: `C1` has disappeared and been replaced by a particular value.

At this point the symbol `y` has been assigned to the symbolic expression on your screen. This is the analytic solution to the initial value problem. What if we want its value at say $x = 2$? To do this we want to (1) bind `x` to 2 in the MATLAB "workspace," and (2) effect this substitution in the expression bound to `y`. The first step is done by typing `x=2`. Still, though, if you type `y` you'll see that this substitition hasn't taken effect. Type `subs(y)` for this. You should get a numerical value for $y(2)$; turn it in as **(c)**. (If you have version 5.0 of the Student Edition, `subs(y)` will return a symbolic expression. To see its value try typing `numeric(ans)`.) To return the symbol `x` to its unbound state, you can type `clear x`.

An alternative approach, which doesn't define the symbol `x` in the workspace, is to type `subs(y,'x',2)`. This substitutes 2 for `x` in the expression `y` alone. (Polking's book is based on an earlier version of MATLAB, and this is one of the places it is out of date: on p. 63 he indicates that one should write `subs(y,2,'x')`, but this is wrong.) Either of these approaches may be used to specify values for the constant of integration, as well.

Second order equations work exactly the same way. Double differentiation is represented by `D2`, so if we are interested in the second order homogeneous linear differential equation $y" + 2y' + 17y = 0$ we could type `de2='D2y+2*Dy+17*y=0'` and then `dsolve(de2,'x')`. The general solution appears, with constants of integration `C1` and `C2`. Notice that MATLAB uses cos and sin rather than complex exponentials. Initial conditions can be entered in the expected way: `y1=dsolve(de2,'y(0)=1','Dy(0)=0','x')` yields the first of a pair of solutions normalized at 0. Type this, and similarly find the solution y_2 such that $y_2(0) = 0, y_2'(0) = 1$, make the results human-readable, and write this on your assignment as **(d)**.

Suppose that we want to graph these solutions. The best MATLAB routine for plotting symbolic expressions is `fplot`. The command `fplot` requires a symbolic expression as input, followed by an interval. To plot the constant function 1 you would not type `fplot(1,[0,1])`: you'll get an error return. The rules concerning symbolic expressions in MATLAB are so arcane, unfortunately, that the best strategy is often trial and error. For example, `fplot('1',[0,1])` works no better, but `fplot('0*x+1',[0,1])` gives a good result. `fplot` looks for a variable name, and will use it as the name of the independent variable.

Now copy the algebraic expression for y_1 given by `dsolve` into the gap in `fplot('   ',[0,6])`. A nice graph should appear. Give it a grid: `grid`. To plot y_2 on the same graph type `hold on` and then the `fplot('   ',[0,6],'r')`, where now the expression for y_2 is inserted. The `'r'` causes the new graph to appear in red. You should be able to observe the damped oscillation, and the alternation of zeros. This pair of solutions is normalized at 0, as you can see. Print this graph out (black and white is fine) and turn it in as **(e)**.