> restart;

Maple Worksheet to Accompany Sections 5.3 of Nagel and Saff

Forced Vibrations

Prepared by Doug Meade
6 April 1995

File: sec5-3.ms

The differential equations that appear in this section are no different from the ones we have studied in Chapter 4. In fact the only difference from the equations earlier in Chapter 5 is the presence of a nonhomogeneous term. Remember that this means it is necessary to find both the homogeneous solution and a particular solution prior to using the initial conditions.

We will talk only about sinusoidal forcing terms. The discussion does not change if the forcing term is something else; it’s just that the sinusoidal forcing term presents some interesting phenomena.

WARNING:
The text contains many formulae in this section. I do not recommend the memorization of ANY of these equations. I don’t know them, and don’t expect you to either.

I do expect that you are familiar with the terms (transient, steady-state, frequencey gain, resonance frequency and resonance curve).

This is a LONG worksheet. However, the individual steps are all very manageable. There is no new Maple and no new mathematics. The only new information is the type of analysis that we perform on the solutions.

Example 1 (Problem #2, p. 216)

```
> ODE := diff( x(t), t,t ) + 2*diff( x(t), t ) + 5*x(t) = 3*cos(4*t);
> IC1 := x(0)=0;
> IC2 := D(x)(0)=0;
> EOM := dsolve( { ODE, IC1, IC2 }, x(t) );
>
```

This is NOT what we would have liked. Can you identify the homogeneous or particular solutions in this expression?

We KNOW the general form of the solution. Let's resort to simpler methods to find the solution in a more tractable form.

First, there is the homogeneous solution:

```
> hSOLN := rhs( dsolve( lhs(ODE)=0, x(t) ) );
>
```

A particular solution is easily found by the method of undetermined coefficients:

```
> guess := A*cos(4*t) + B*sin(4*t);
> GOAL := simplify( subs( x(t)=guess, ODE ) );
> EQNS := { -11*A + 8*B = 3, -11*B - 8*A = 0 };
> COEFFS := solve( EQNS, { A, B } );
> pSOLN := subs( COEFFS, guess );
>
```

The general solution to the ODE is now seen to be

```
> gSOLN :=  hSOLN + pSOLN;
>
```

The initial conditions can be used to find the constants _C1 and _C2.

```
> EQNS := { simplify( subs( t=0, gSOLN ) ) = 0, simplify( subs( t=0, diff( gSOLN, t ) ) ) = 0 };
> CONST := solve( EQNS, { _C1, _C2 } );
> SOLN := subs( CONST, gSOLN );
>
```

That’s better. But, we are not quite finished. To fully appreciate this solution, note that we can combine the pairs of sine and cosine in the homogeneous and nonhomogeneous solutions, respectively. After a little work, the solution can be written as:

```
> SOLNa := exp(-t) * 3*5/sqrt(37)/10 * sin( 2*t + 2.33294 ) + 3/sqrt(185) * sin( 4*t - 0.942 );
>
```

NOTE: BE SURE YOU KNOW HOW I DETERMINED THIS SOLUTION!

If this is correct, then there should be only one curve in the followng plot.

```
> plot( { SOLN, SOLNa }, t=0..2, title=‘Do the two solutions agree‘ );
>
```

To see the different components of this solution, let’s consider a few more graphs. First, let’s look at the solution by itself.

```
> plot( SOLN, t=0..10, title=‘Solution to the IVP‘ );
>
```

Notice that the first few oscillations look different from the later oscillations, which are all the same. This is due to the fact that the homogeneous part of the solution decays exponentially and the nonhomogeneous part continues to oscillate with no change in its amplitude.

The part of the solution that decays is call the TRANSIENT SOLUTION. The portion of the solution that persists forever is the STEADY-STATE SOLUTION.
Let’s plot the two parts separately.

```
> TRANSIENT := subs( CONST, hSOLN );
> SS := subs( COEFFS, pSOLN );
> plot( { TRANSIENT, SS }, t=0..10, title=‘Transient and Steady-State Solutions‘ );
>
```

Now, go back and add the full solution to this plot.

Next, let’s compare the steady-state solution with the forcing term.

> plot({ SS, rhs(ODE) }, t=0..10, title=‘Steady-State Solution and Forcing Term‘);
>

Notice that the amplitudes are different, but the periods of the two functions are the same (approximate). The two functions are also out of phase by a fixed amount (approximate). Note the relation between these quantities and the steady-state solution.

The last piece of information that we will extract from this solution is the FREQUENCY GAIN (the ratio of the amplitudes of the steady-state solution and the forcing term. Here, this is:

> M := 3/sqrt(185) / 3; evalf(M);
>

EXTENSION -- RESONANCE AND FREQUENCY RESPONSE CURVES

The frequency gain will be change as the frequency of forcing term changes. For example, in the above example we found that an input frequency of 4/(2*Pi) = 2/Pi produces a frequency gain of 0.0735. Reducing the input frequency to 2/(2*Pi) = 1/Pi increases the frequency gain to 0.2425 and an input frequency of 1/2/(2*Pi) = 1/(4*Pi) has a frequency gain of 0.20601.

The input frequency that maximizes the frequency gain is called the RESONANCE FREQUENCY.

WHAT IS THE RESONANCE FREQUENCY FOR THIS SYSTEM?
WHAT DOES THE RESONANCE SOLUTION LOOK LIKE?

To answer these questions it is necessary to consider the problem with a VARIABLE frequency in the forcing term. That is, consider the IVP:

```
> constants := op( {constants} minus {gamma} ):
> ODE := diff( x(t), t,t ) + 2*diff( x(t), t ) + 5*x(t) = 3*cos(gamma*t);
> IC1 := x(0)=0;
> IC2 := D(x)(0) = 0;
>
```

The solution to this system can be found exactly as above:

```
> hSOLN := rhs( dsolve( lhs(ODE)=0, x(t) ) );
> guess := A*cos(gamma*t) + B*sin(gamma*t);
> GOAL := simplify( subs( x(t)=guess, ODE ) );
> EQNS := { -gamma^2*A + 2*gamma*B + 5*A = 3, -gamma^2*B - 2*gamma*A + 5*B = 0 };
> COEFFS := solve( EQNS, { A, B } );
> pSOLN := subs( COEFFS, guess );
> gSOLN :=  hSOLN + pSOLN;
> EQNS := { simplify( subs( t=0, gSOLN ) ) = 0,
            simplify( subs( t=0, diff( gSOLN, t ) ) ) = 0 };
> CONST := solve( EQNS, { _C1, _C2 } );
> SOLN := subs( CONST, gSOLN );
>
```

What is the frequency gain for this solution?

Well, it's just the ratio of the amplitudes of the steady-state solution and the forcing term (which is 3).

```
> SS := pSOLN;
> A_SS := simplify( sqrt( (-3*(gamma^2-5)/(25-6*gamma^2+gamma^4))^2 + ( 6*gamma/(25-6*g
  amma^2+gamma^4) )^2 ) );
> M := A_SS / 3;
>
```

Let's pause and check that this is consistent with the numbers reported earlier:

```
> evalf( subs( gamma=4, M ) );
> evalf( subs( gamma=2, M ) );
> evalf( subs( gamma=1/2, M ) );
>
```

Now, how can we find the maximum value of the frequency gain, and the resonance frequency?

```
> plot( M, gamma=0..10, title='Frequency Response Curve' );
> dM := diff( M, gamma );
> dM := simplify( dM );
> CRITPTS := solve( dM=0, gamma );
> allvalues( [ CRITPTS ] );
>
```

Thus, the frequency gain is maximized when gam=sqrt(3), i.e. the resonance frequency is

```
> sqrt(3)/(2*Pi); evalf( " );
>
```

the maximum frequency gain is

```
> simplify( subs( gamma=sqrt(3), M ) );
>
```

and the resonance solution is

```
> resSOLN := subs( gamma=sqrt(3), SOLN );
>
```

To conclude, compare the resonance solution with the solutions with three different input frequencies.

```
> SOLNa := subs( gamma=4, SOLN );
> SOLNb := subs( gamma=2, SOLN );
> SOLNc := subs( gamma=1/2, SOLN );
> plot( { resSOLN, SOLNa, SOLNb, SOLNc }, t=0..10, title='Which Curve is the Resonance Sol
  ution?' );
>
```

How do you know which solution corresponds to the different values of gam?

The behavior of the resonance solution is different if the motion is undamped. In this case the amplitude of the resonance solution increases with time. For more information on this case, please see the discussion in the text (pages 214--216).

END OF WORKSHEET

```
>
```