CSC260S: Intro to Scientific, Graphical and Symbolic Computing
Department of Computer Science
University of Toronto

Instructor: E. Fiume **Term Test** **24 Feb. 1995**

Closed book. The test questions continue overleaf. Show your work. Work quickly.

1. [10] We are given an arbitrary polynomial $p(t)$ that is of interest to us on the interval $[0,1]$.

(a) [2] Show (mathematically) how to reparameterise $p(t)$ to an identically-behaving polynomial $\hat{p}(u)$ on an arbitrary interval $[a,b]$.

If $t \in [0,1]$, then $u = a + t(b-a) \in [a,b]$. We form the substitution $t = \frac{u-a}{b-a}$. Our reparameterised polynomial is therefore $\hat{p}(u) = p\left(\frac{u-a}{b-a}\right)$.

(b) [3] Reparameterise the polynomial $p(t) = (t - \frac{1}{2})(t - \frac{3}{4})$ from [0,1] to [−1,1], expressing the result in monomial form in a new parameter u.

(I specified the polynomial this way to make the reparameterisation easier.)

```
> repar((t-1/2)*(t-3/4), -1, 1, u);
                                        2
                              1/4 u  - 1/8 u
```

(c) [5] Write a Maple procedure **`repar(p,a,b,u)`** that performs the reparameterisation from (a), taking a polynomial **`p`** in an indeterminate that you must extract, and returning a new polynomial in **`u`** parameterised on **`[a,b]`**. Your Maple code should perform reasonable error checking on the input domain and polynomial **`p`**. The result returned should be in monomial form in **`u`**.

Something like this would be fine. There isn’t much "real" code here.

```
repar := proc(p,a,b,u)
     local X,q;
     X := indets(p);
     if (nops(X) <> 1) then
          ERROR(‘incorrect number of indeterminates in polynomial‘);
     fi;
     if (a >= b) then
          ERROR(‘incorrect domain‘);
     fi;
     q := subs(X[1] = (u-a)/(b-a), p);
     collect(q,u);
end:
```

2. [3] Consider the parametric curve $\mathbf{F}(t) = (x(t), y(t)) = (t-1, t^2-1)$. Convert $\mathbf{F}$ into an *explicit* function of the form $y = f(x)$.

$$y = (t-1)(t+1) = x\cdot(t+1) = x\cdot((t-1)+2) = x\cdot(x+2) = x^2 + 2x.$$

3. [5] Sketch an (x,y) plot of each of the following parametrically defined curves given by $\mathbf{F}(t) = (x(t), y(t))$. In each case, $t \in [-1,1]$. Your sketch need not be a masterpiece, but it should indicate the endpoints accurately.

Use Maple to plot these.

(a) [2] $\mathbf{F}(t) = (1, t^3 + 3t^2 - 1)$.

A vertical line segment at $x = 1$ with $-1 \le y \le 3$. Note: the minimum of -1 is achieved when $t=0$, which made this question harder than expected. We'll be kind.

(b) [3] $\mathbf{F}(t) = (t - 1, t^2 - 1)$.

A segment of a parabola. (Use previous question to help you out, or otherwise just plot points.)

4. [10] The parametric definition of an ellipse centred at the origin is

$$x(u) = a \cos u,$$
$$y(u) = b \sin u,$$

for positive $a, b \in \mathbf{R}$, and for $u \in [0, 2\pi]$. We would like to compute a sequence of points $\left((x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n) \right)$ on the ellipse so that

$$(x_i, y_i) = \left(x(i\Delta u), y(i\Delta u) \right),$$

for some fixed Δu, and starting with $(x_0, y_0) = (a, 0)$.

(a) [4] Write a Turing or C fragment with appropriate definition of variables that directly outputs these points (using **`put`** or **`printf`**).

```
var x,y: real
const du:  real := {something}
const n:   int  := {something}       % should input these values
const a,b: real := {something}

x := a
y := 0
for i:0..n
    put "(",x, ", ", y, ")"
    x := a*cos(i*du)
    y := b*sin(i*du)
end for
put "(",x, ", ", y, ")"
```

(b) [6] Unfortunately, the machine on which we will be running this program cannot compute sines and cosines. Derive an incremental form for computing the sequence of points so that the computation is of the form

$$x_{i+1} \approx x_i + x'(u)\Delta u,$$
$$y_{i+1} \approx y_i + y'(u)\Delta u,$$

but containing *no* sine or cosine functions. Here, x' and y' denote the derivative of x and y with respect to u. Then rewrite your code fragment with this derivation. Don't worry about the possible numerical instability of the result.

Taking derivatives, we see that

$$x'(u) = -a \sin u = -\frac{a}{b} y,$$

$$y'(u) = b \cos u = \frac{b}{a} x.$$

Our iteration can therefore be written as

$$x_{i+1} := x_i - \frac{a}{b} y_i \Delta u,$$

$$y_{i+1} := y_i + \frac{b}{a} x_i \Delta u.$$

Careful! You shouldn't destroy your current x value when converting this to code. By a happy coincidence, however, if you forget to save the old value of x, this results in a **better** algorithm. This is a very rare case in which a bug gives you a better implementation.

```
var x,y: real
const du:  real := {something}
const n:   int  := {something}       % should input these values
const a,b: real := {something}

x := a
y := 0
u := du
const cx := -a/b*du
const cy :=  b/a*du
for i:0..n
    const savex := x     % a "Turing convention"
    put "(",x, ", ", y, ")"
    x += cx*y
    y += cy*savex            % the "pure" form of iteration
    x  = savex
end for
put "(",x, ", ", y, ")"
```

5. [12] Lagrange interpolation takes a sequence $\big((t_0,x_0),(t_1,x_1),\cdots,(t_n,x_n)\big)$ of points and produces a polynomial $p(t)$ that interpolates the x data values in the given order.

This is easy going until 5d.!

(a) [2] Write down the interpolation constraints that $p(t)$ must satisfy, in terms of the input sequence.

$$p(t_i) = x_i, \; i = 0, \cdots, n.$$

(b) [2] Suppose my data is of the form $x_i = mt_i + b$ for each i, for fixed values m and b. What kind of curve would you expect the resulting $p(t)$ to be?

We certainly expect the result to be a line, and more specifically,

$$p(t) = mt + b.$$

Now suppose we wish to interpolate points using the four-point interpolant, (or equivalently, using Lagrange interpolation), on four points, A, B, C, D, spaced uniformly with respect to a parameter $t \in [0, 1]$. The result is a polynomial space curve $\mathbf{p}(t)$ that interpolates these points in order as t varies from zero to one.

(c) [3] Write down a set of equations that define the interpolation constraints that must be true of $\mathbf{p}(t)$.

$$\begin{aligned} \mathbf{p}(0) &= A, \\ \mathbf{p}(1/3) &= B, \\ \mathbf{p}(2/3) &= C, \\ \mathbf{p}(1) &= D. \end{aligned}$$

(d) [5] Suppose I try to interpolate the sequence of points A, A, B, B. Can we expect the resulting curve $\mathbf{p}(t)$ to be as in (b)? If so, argue why this is the case. If not, sketch the curve for an example of A and B in $\mathbf{R}^2$, and argue why the curve is not like (b).

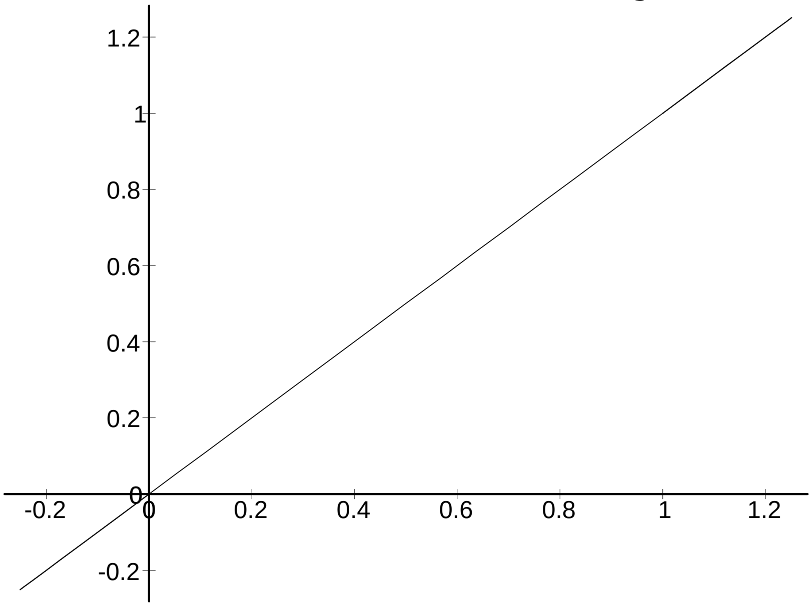
Note: my first answer to this was incorrect. My thanks to a student for noticing this. I will go through this carefully, because it's really fascinating (to me, anyway). The result *is* in fact a straight segment, but *not* the same straight segment as in (b) above, in the sense that the one you get in (d) generally strictly *includes* the segment got from (b). You'll see what I mean by an example. The four-point cubic interpolating only two points can be written in a procedure like so.

```
plotcubic := proc(A,B,t0,t1)
    local x,y,t;
    x :=interp([0,1/3,2/3,1],[A[1],A[1],B[1],B[1]],t);
    y :=interp([0,1/3,2/3,1],[A[2],A[2],B[2],B[2]],t);
    print(plot([x,y,t=t0..t1]));
end:
```

When I (for example) execute

```
plotcubic([0,0],[1,1], 0,1);
```

this should give me a plot of the resulting space curve from (0,0) to (1,1) when t goes from 0 to 1. But here's the plot.



Observe that in fact the line segment overshoots (0,0) and (1,1). What I said originally about the "curve looping" is true: it's just that the "loop" is squeezed onto the line segment. Now, how would we prove that the result is a line segment in general?

We can use Maple to do it. The following procedure will compute the polynomial space curve interpolating arbitrary points A,B:

```
interpcubic := proc(A,B,t0,t1)
    local x,y;
    x :=interp([0,1/3,2/3,1],[A[1],A[1],B[1],B[1]],t);
    y :=interp([0,1/3,2/3,1],[A[2],A[2],B[2],B[2]],t);
    [x,y];
end:
```

Now, the general implicit equation for a line in space is

$$L: \; ax + by + c = 0,$$

for constants $a,b,c \in \mathbf{R}$. So, if we can find constants a,b,c such that

$$ax(t) + by(t) + c = 0$$

for any t, we're done. There are lots of ways to do this, but we can use Maple directly. I won't include too much Maple output because it's messy. Please try this out. First, define our endpoints and the corresponding curve passing through them.

```
> A := [x0,y0]:
> B := [x1,y1];
> P := interpcubic(A,B,0,1):
> X := P[1]:
> Y := P[2]:
```

So, now X and Y are polynomials. Let's form a "possible" line segment and see if we can solve for the coefficients. There are three coefficients, so let's just evaluate the space curve at three points, use maple to solve for the three coefficients, and see if that solution for a,b,c will do. Here goes:

```
> L := a*X + b*Y + c:
> L1 := subs(t=0,L):
> L2 := subs(t=1/2,L):            # evaluate at three points.
> L3 := subs(t=1,L):
> solve({L1,L2,L3},{a,b,c});      # solve L1=0,L2=0,L3=0 for {a,b,c}

              (- y1 + y0) b         b (x0 y1 - y0 x1)
       {a = - -------------, c = - -----------------, b = b}
                - x1 + x0               - x1 + x0

> simplify(subs(",L));

                          0
```

Thus the solution above works for any choice of b. (Note, I could have sampled at 0,1/3,2/3,1, which would make life easier if I were doing this by hand. Why?)

So my original answer was heading in the right direction but was not entirely correct. It still would be an acceptable answer to this question. It is also acceptable to say it is a line segment, but it is not the same line segment as in (b). Of course only an intuitive argument would be needed in either case.