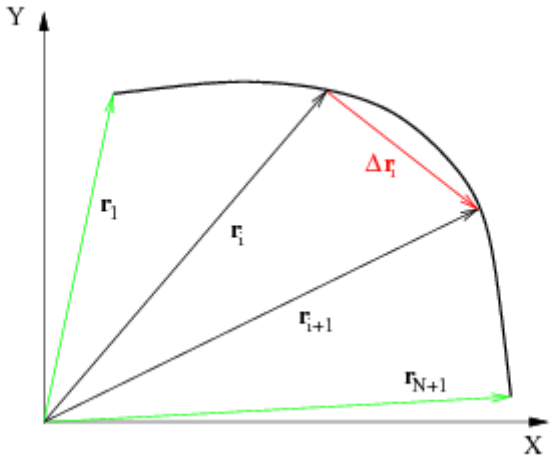
<texit info> author=Roman Putanowicz title=Solution to exercise 6.1.2

</texit> back

Solution to exercise 6.1.2

The N segments that partition the parameter space are defined by values \begin{equation*} \{t_1, t_2,\ldots,t_{N+1}\}\enspace . \end{equation*} For each value of t_{i} we can calculate the corresponding radius vector \begin{equation*} \mathbf{r}_i = \mathbf{r}(t_i) = [x(t_i), y(t_i)] \enspace . \end{equation*} Then the straight line segments that approximate the curve can be defined as \begin{equation*} \Delta\mathbf{r}_i = \mathbf{r}_{i+1} - \mathbf{r}_i \quad \text{for}\quad i=1,\ldots,N \enspace. \end{equation*}



Adding the lengths of vectors give us the desired approximation of the curve length: \begin{equation*} S = \sum_{i=1}^{i=N} ||\Delta\mathbf{r}_i||\enspace . \end{equation*}

There are, in general, two ways of implementing the above formula in Octave. The first is to use Octave ability to operate on vectors and matrices and some of Octave built-in functions like **norm** for calculating vector length or **sum** for calculating sum of vector elements. By doing this one can avoid using explicit loop instructions in Octave and this account for faster scripts. More importantly the script will be closer to the mathematical notation thus in most cases less error prone.

Below is the script using the vector approach: <sxh c> function r = curve(t)

```
r = zeros(2, length(t));
r(1,:) = 2*cos(t);
r(2,:) = 2*sin(t);
```

endfunction

ts = input(“Give start point parameter, ts : ”); te = input(“Give end point parameter, te : ”); N = input(“Gvie number of segmetns, N : ”);

t = linspace(ts, te, N+1);

r = curve(t);

dr = diff(r, 1, 2);

s = sum(norm(dr,2,“cols”));

printf(“Curve length : %g\n”, s); </sxh>

<texit> \begin{lstlisting} function r = curve(t)

```
r = zeros(2, length(t));
r(1,:) = 2*cos(t);
r(2,:) = 2*sin(t);
```

endfunction

ts = input(“Give start point parameter, ts : ”); te = input(“Give end point parameter, te : ”); N = input(“Gvie number of segmetns, N : ”);

t = linspace(ts, te, N+1);

r = curve(t);

dr = diff(r, 1, 2);

s = sum(norm(dr,2,“cols”));

printf(“Curve length : %g\n”, s); \end{lstlisting} </texit>

Depending on the size of the argument `t` the funtion `curve` returns one column vector being the radius vector for `t` or a matrix, such that each column of it is a radius vector for respective `t`. In line 11 we calculate vector `t` whose elements are distributed uniformly in the range [`ts`, `te`]. In line 15 we use **diff** function to calculate matrix of the first differences between columns of matrx `r`. In line 17 in turn, we request to calculate the norms of each column of matrix `dr`.