2/5/96

# TI-82 programs for numerical integration

## Information about the programs

These are straightforward programs designed to let students easily compute three numerical approximations to definite integrals. They are not designed to be production-perfect software, so there is no error checking, and minimization of roundoff error and maximization of efficiency have not been considered. What was desired were three programs which would be direct and almost readable, and which would "work". The programs do indeed "work" and do provide rough answers to the question, "What happens when you apply the XYZ RULE to estimate the integral ...?" Each program is typed into the calculator by beginning with the `PRGM` key followed by the choice of `NEW`. Each program can then be used by pressing the `PRGM` key followed by `EXEC` (execute). Read chapter 13 of the TI-82 guidebook for more information.

`RECT` uses right-end endpoints of subdivisions and areas of rectangles to approximate a definite integral.

`TRAP` uses the Trapezoid Rule to approximate a definite integral.

`SIMP` uses Simpson's Rule to approximate a definite integral.

Each program can be used in the same way. For each of them, the function to be "integrated" is entered into $\mathtt{Y}_1$ (the first function under the control of the `Y=` key).

In each case, the program asks for `L` and `R` and `N`. The program assumes that we are trying to approximate the definite integral of $\mathtt{Y}_1$ from `L` (the left-hand endpoint) to `R` (the right-hand endpoint): $\int_L^R Y_1(x)\,dx$. Here `N` is the number of subintervals desired. `N` must be even for Simpson's Rule which is implemented in the program `SIMP` below.

```
:Prompt L,R,N
```

In each program `S` is the sum, that is, the value of the approximation to the desired definite integral. Each program sets `S` to `0` initially. The symbol → is an abbreviation for the result of pressing the STO (store) key.

```
:0→S
```

Each program then computes `W`, the subinterval widths, so this computation is not done repeatedly later.

```
:(R-L)/N→W
```

Each program then uses the `For( , , , )` construction. The instructions of the program are executed repeatedly between the `For` and the `End` instruction. In the first two programs (`RECT` and `TRAP`) the "loop" (the collection of instructions between `For` and `End`) is executed `N` times, and the variable `A` is incremented by `1` on each trip through the loop. The value of `A` is used to select right-end endpoints of each subinterval, and then to create a suitable addition to the sum, `S`. In the program `SIMP` the variable `A` is incremented by 2 during each trip through the loop, and the two subinterval points are "weighted" differently (one multiplied by 4 and the other multiplied by 2).

`:For(A,1,N,1)` or `:For(A,1,N,2)`
(some instruction changing `S`)
`:End`

The program then either displays the sum and stops, as this quote from `RECT` indicates:

```
:Disp S
:Stop
```

or (in `TRAP` and `SIMP`) a small amount of further computation is done to "fix up" `S` with the values of $\mathtt{Y}_1$ at the endpoints (there's also a little more fixing up for `SIMP`), followed by a display of the sum and the end of the program.

**Please turn over for the programs.**

2/5/96

# TI-82 programs for numerical integration

### The programs themselves

```
PROGRAM:RECT
:Prompt L,R,N
:0→S
:(R-L)/N→W
:For(A,1,N,1)
:S+Y1(L+A*W)*W→S
:End
:Disp S
:Stop
```

```
PROGRAM:TRAP
:0→S
:Prompt L,R,N
:(R-L)/N→W
:For(A,1,N,1)
:S+Y1(L+A*W)*W→S
:End
:S+((.5*Y1(L))*W)-((.5*Y1(R))*W)→S
:Disp S
:Stop
```

```
PROGRAM:SIMP
:0→S
:Prompt L,R,N
:(R-L)/N→W
:For(A,1,N,2)
:S+(4*Y1(L+A*W)*W)+(2*Y1(L+(A+1)*W)*W)→S
:End
:S+Y1(L)*W-Y1(R)*W→S
:S/3→S
:Disp S
:Stop
```

Note that some of the longer lines written above will “fold over” on the TI-82’s screen. They are entered, though, without paying any attention to this phenomenon.

**Please turn over for information about the programs.**