CSCI 551
Numerical and Parallel Programming

Lecture 5-2 – Introduction to MPI and Exercise #3 Help

February 17, 2025

MPI Simple Demonstrations

- MPI Hello world
  - `mpicc -g -Wall -o greetings greetings.c`
  - `mpirun -n 4 ./greetings`
- Use starter Makefile in CSCI 551 code directories!
- MPI book examples
  - Always works mpiexec
  - Using mpirun on multiple nodes can hang (program error) OR due to cluster node issue

```
sbsiewert@o244-01:~/code/hello_custer$ ls
c1_machine_file  c2_machine_file  greetings  greetings.c  Makefile
sbsiewert@o244-01:~/code/hello_custer$ mpirun -n 4 ./greetings
Hello from process 0 of 4 on o244-01
Hello from process 1 of 4 on o244-01

Hello from process 2 of 4 on o244-01

Hello from process 3 of 4 on o244-01

sbsiewert@o244-01:~/code/hello_custer$
```

```
4. ecc-linux (sbsiewert)
sbsiewert@o251-01:~/public_html/csci551/code/MPI_Examples$ make
gcc -g -Wall -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/include/  -o
debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib
gcc -g -Wall -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/include/  -o
lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib
gcc -g -Wall -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/include/  -o
l64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib
gcc -g -Wall -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/include/  -o
g -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
nux/mpi/intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
0.166/linux/mpi/intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
0.166/linux/mpi/intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
x/mpi/intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/l
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
i/intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
linux/mpi/intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib -lm
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib -lm
mpicc -g -Wall -O2 -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/includ
intel64/lib/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib -lm
gcc -g -Wall -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/include/  -o
b/debug -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib -lm
sbsiewert@o251-01:~/public_html/csci551/code/MPI_Examples$ ls
bubble      ex4vel.h   mpi_hello.c      mpi_mat_vect_mult    mpi_mat_vect_time.c  mpi_outpu
bubble.c    Makefile   mpi_many_msgs    mpi_mat_vect_mult.c  mpi_odd_even         mpi_outpu
ex4accel.h  mpi_hello  mpi_many_msgs.c  mpi_mat_vect_time    mpi_odd_even.c       mpi_trap1
sbsiewert@o251-01:~/public_html/csci551/code/MPI_Examples$ mpiexec ./mpi_hello
Greetings from process 0 of 4!
Greetings from process 1 of 4!
Greetings from process 2 of 4!
Greetings from process 3 of 4!
sbsiewert@o251-01:~/public_html/csci551/code/MPI_Examples$
```

Exercise #3

- **Read chapter 3 through p. 113 – Intro to MPI**
- **Pacheco Code - csci551/code/MPI_Examples/**
  - **Integrate any function – code walkthrough of math function and interpolation methods**
  - **csci551/code/MPI_Examples/mpi_trap3.c**
  - **csci551/code/MPI_Examples/mpi_trap4.c**

- Review LLNL tutorials (for more on Pthreads as you are interested)
- Review Intel MPI benchmark user's guide (csci551/code/mpi-benchmarks-master-csu/) – Broken now … ignore

- Simple MPI program – based on book to explore MPI and cluster features

- Top Down - MPI benchmarks and examples
  - Build, run, observe
  - Review code and explain

- Bottom Up – Methods of integration
  - Simple forward integration, large, medium, small step size
  - Trapezoidal integration, large, medium, small step size

- Application – Train Acceleration and Braking
  - Create a profile of position and velocity over time
  - Given acceleration timeline
  - Compute arrival time
  - Use MPI to speed up the analysis (6 runs, "n" segments to integrate)

Notes on Simulation

- Functions with known anti-derivative simpler (Ex #3)
  - We know acceleration, velocity, and even position functions
  - Integration here can be verified with symbolic math
- Function Look-up Table with Interpolation (Ex #4)
  - Review of interpolation - models any function
  - Harder loop-carried dependency - csci551/code/functiongen/
- Discussion of Simulation and Parallel Divide in Conquer
  - OpenMP version with speed-up
  - Phases of integration
- Checking correctness for integration
  - Ex-3 model - Ex-3-Smooth-Function-Train-profile.xlsx
  - Ex-4 model - Ex-4-Segmented-Function-Train-profile.xlsx

Exercise #3 & #4 – MPI for Simulation

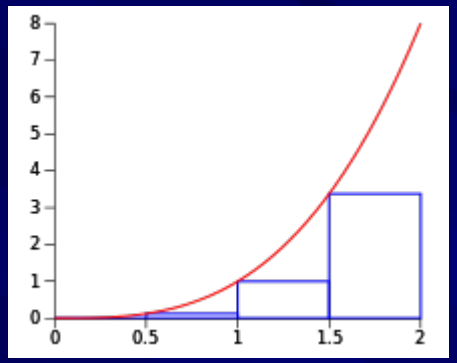
- Continuous physics of 1-DOF translation
  - Position = P_0 + Velocity x time, if V(t) is constant
  - Velocity = V_0 + Acceleration x time, if A(t) is constant
  - Acceleration can be constant, or could be a function of time

- If Acceleration is f(t), we need to integrate
  - Simple Riemann Sum Integration
  - Position(t) = $P_0 + \sum V(t)dt$
  - Velocity(t) = $V_0 + \sum A(t)dt$
  - Acceleration(t) = f(t), provided as acceleration profile (or from sensor)

- Approximate the area under curve A(t) for a dt (step size)
- Break A(t) into segments or make it a look-up-table
- Interpolate for A(t) at times between profile given
- Consider accuracy (method of integration & step size) and precision (double or float)

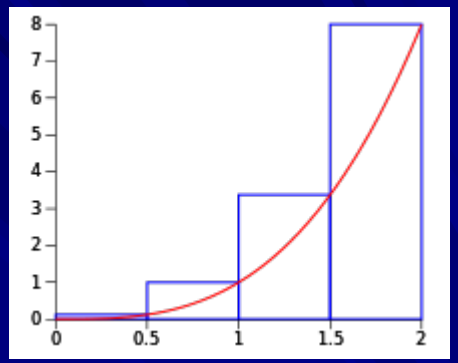
Recall Integration Methods (Riemann Sum)

Left Riemann Sum (Forward)
A_n= f(a) dx
dx=[b-a]/n



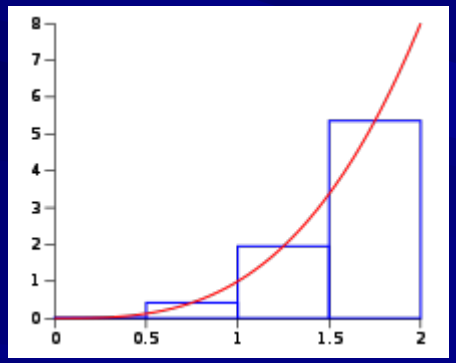
A_{b-a} = dx [(f(a) + f(a+dx) + f(a+2dx) + … + f(b-dx)]

Right Riemann Sum (Backward)
A_n= f(a+dx) dx
dx=[b-a]/n



A_{b-a} = dx [f(a+dx) + f(a+2dx) + … + f(b)]

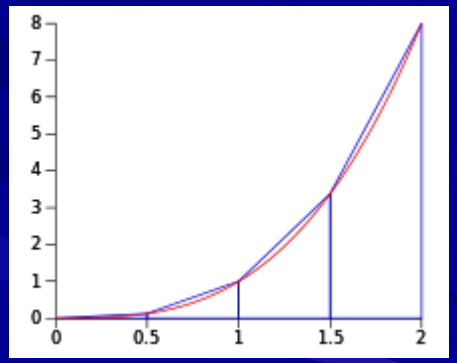
Midpoint Rule
A_n= f(a+dx/2) dx
dx=[b-a]/n



A_{b-a} = dx [f(a+dx/2) + f(a+3dx/2) + … + f(b-dx/2)]

Trapezoidal Rule

A_{trap}=½ h (f(a)+f(b))
dx=[b-a]/n
A_n=½ dx (f(a)+ f(a+dx))



½ dx [(f(a) + f(a+dx)) + (f(a+dx) + f(a+2dx)) …]
½ dx [f(a) + 2f(a+dx) + 2f(a+2dx) + … f(b)]

Numerical integration works for any function – solved by symbolic calculus or not
Can also work on well-defined functions that have known symbolic solutions

Verifying Numerical Integration

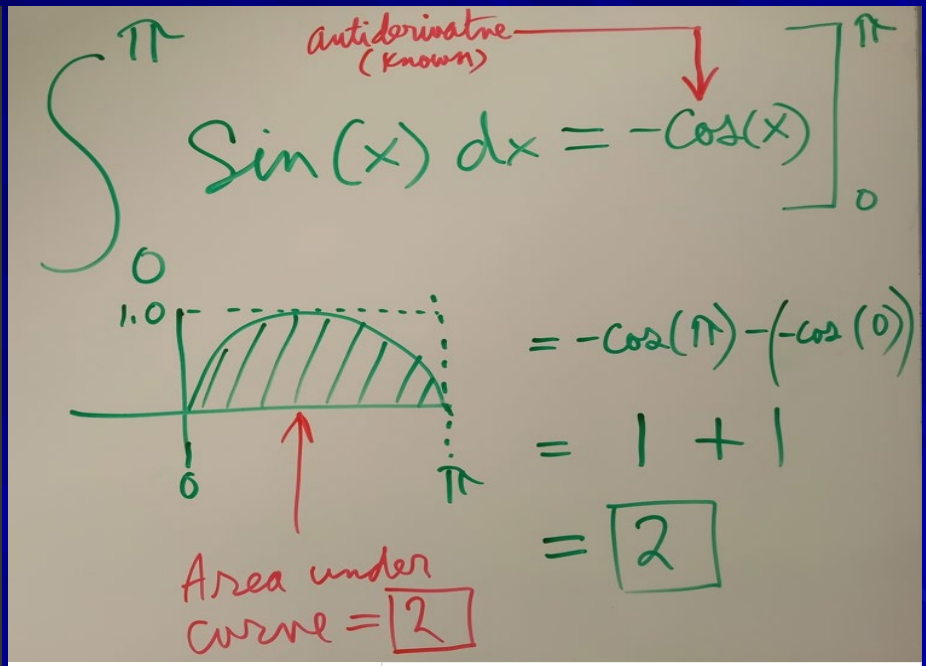
- Test with definite integral from Calculus – exact solution
- E.g., area under Sin(x) over well-defined interval (2.0)

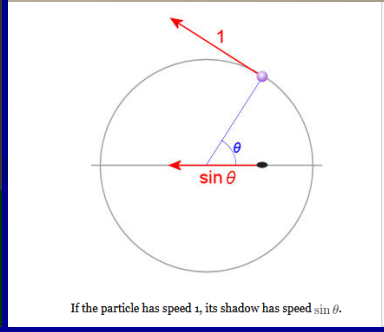
Known anti-derivative of sine and evaluation
- Definite integral solution
- We know from calculus
- Can be proven

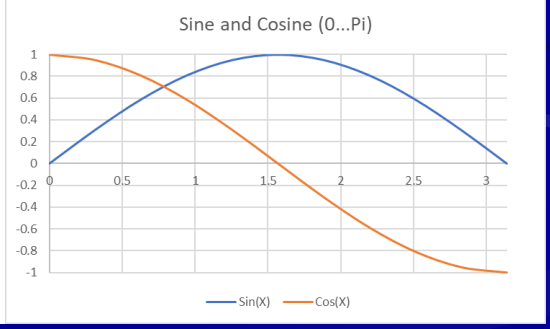
Use this fact to verify your numerical integration
- Riemann Sum (function)
- Trapezoidal (function)
- Simpson's Rule (function)
- Runge-Kutta (differential equations)
- Other methods of quadrature (math topic)
- Estimation of area under a curve (function)
- Integration in general of a differential equation
- In applications, the functions generally represent physical quantities, the derivatives represent their rates of change, and the differential equation defines a relationship between the two.

Unit Circle integration of velocity of shadow
- Assume light source on vertical axis
- Velocity of the shadow is sine(theta)
- Position is the Integral of velocity
- Distance is 2.0 for unit circle

Sine and Cosine Unit Circle Calculus Videos (diameter=2.0):
Unit Circle – Derivative, Integral of Sine(x) Video
https://betterexplained.com/articles/integral-sinx/



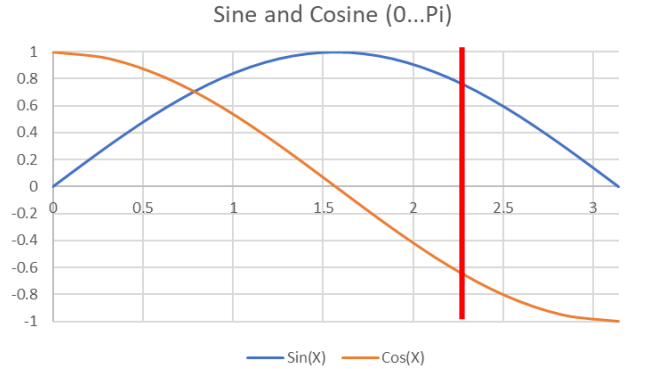


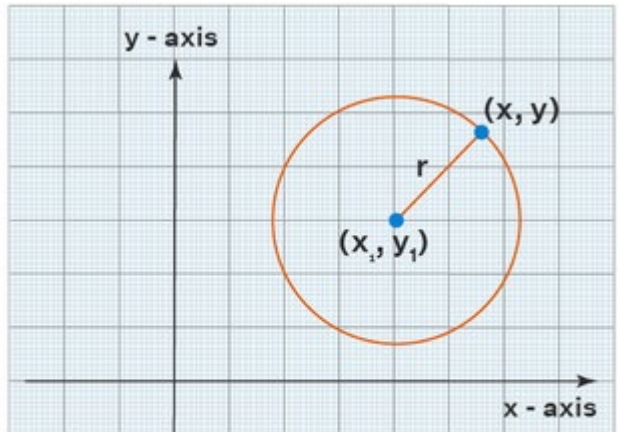


Integration Tips – Stick to Functions

- Keep it Simple! If possible.
- Remember, we are just estimating the area under a curve (function) for most problems
- Recall the definition of a function
  - A vertical line intersects only ONE value of a function y=f(x)
  - Therefore, a circle is not a function, but it is a shape that can be described by an equation
- For the train problems, stick to integration of simple functions if possible
- For Ex #3, a function to model Acceleration(t) will have to be piecewise
- Initial conditions for divide and conquer are a challenge

Function – one vertical intersection



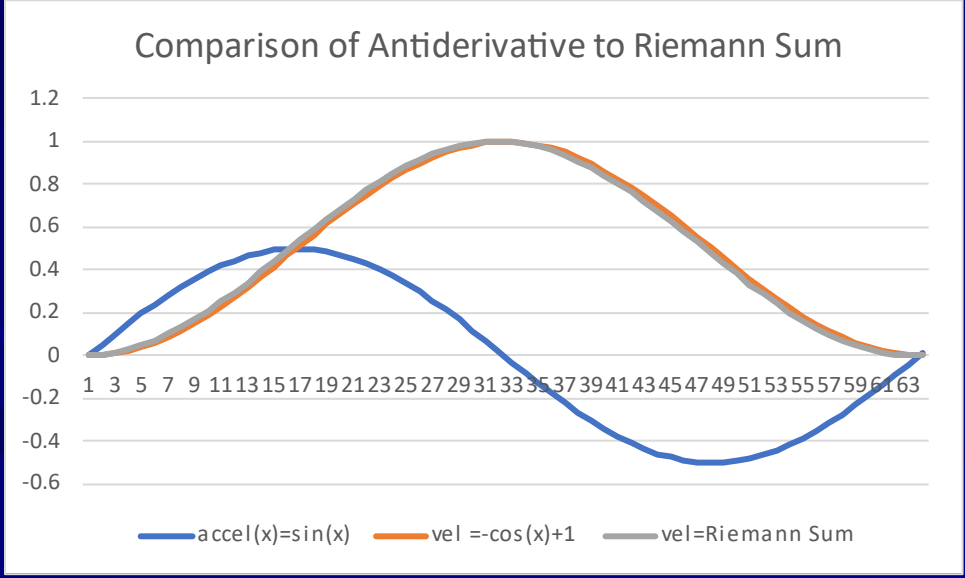


The standard equation of a circle gives precise information about the center of the circle and its radius and therefore, it is much easier to read the center and the radius of the circle at a glance. The standard equation of a circle with center at (x_1, y_1) and radius r is $(x - x_1)^2 + (y - y_1)^2 = r^2$, where (x, y) is an arbitrary point on the circumference of the circle.

https://www.cuemath.com/geometry/equation-of-circle/

Ex #3 Help

- Train Problem is Non-Linear, but Simple sin(x) acceleration function (known from calculus)
- Velocity has a definite integral solution of –cos(x)+C
- Table Look-Up or interpolation (not required)
- Re-Try MPI implementation for n=1 … m processes
- Focus on Accuracy and Precision rather than speed-up



Antiderivative is Exact Solution (Spreadsheet)

Riemann or Trapezoidal is Numerical Approximation

Train acceleration profile is simple sine curve, but scaled for time and amplitude

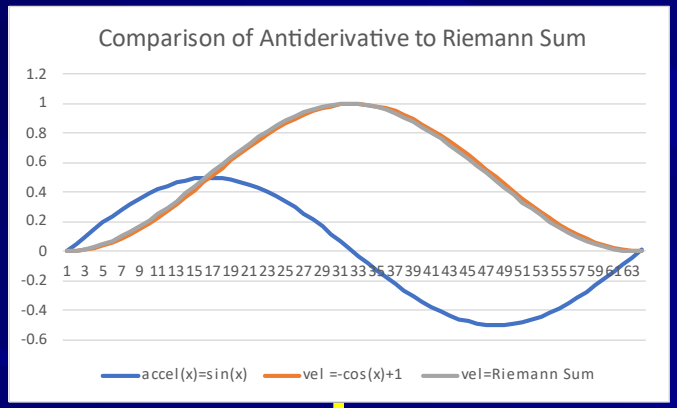
Non-Linear Functions

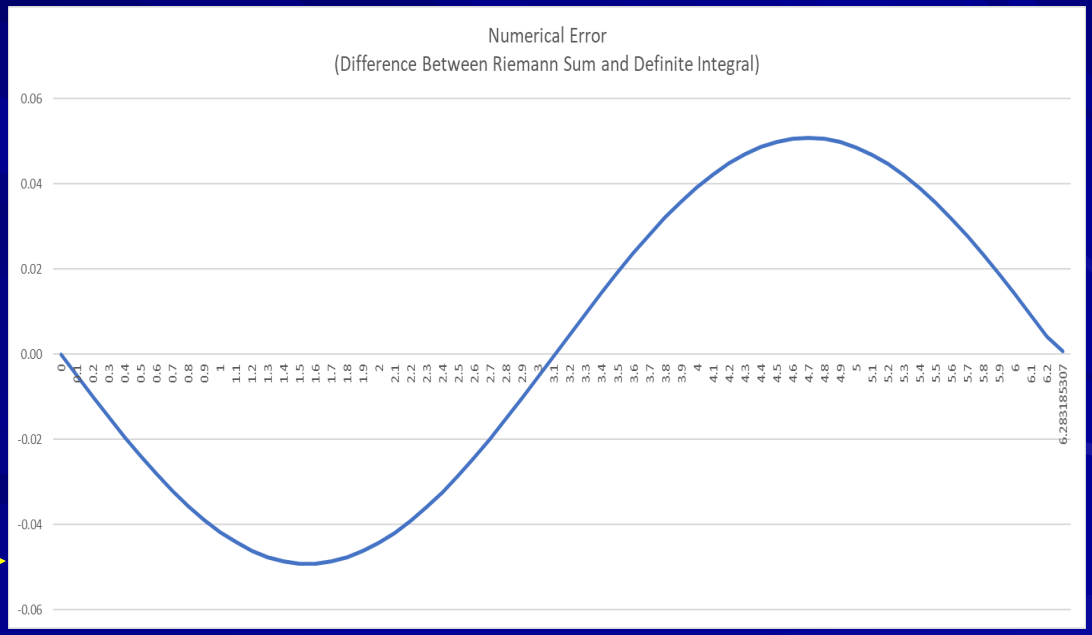
- Polynomials of degree 2 or higher
- Transcendental functions (trigonometric)
- Others (e.g., logarithmic), but these 2 are our primary interest
- Non-linear can be treated as “piece-wise linear”, but often can be directly implemented and solved with mathematics – e.g., antiderivative of sin(x)

- Linear systems include linear functions as well as linear systems of equations
- Non-linear functions and non-linear systems of equations (interesting, but we will keep it simple)
- Ex #3 and Ex #6 are the main practice we have

Most of the world is non-linear

- The world is mostly non-linear, engineers and scientists like to simplify with linear models (do you agree?)
- Potential accuracy issues – e.g., linear interpolation with non-linear functions (piecewise linear, small steps)
- In Ex #4, use direct function generator rather than linear interpolation





Complex Functions

- Most measured functions for real-world applications are hard to model with a simple function and math library
  - Measured acceleration with accelerometer
  - Output is Acceleration(t) as data values in a table
  - Create look-up table and interpolate to find values between measurements
- Look-up tables preclude use of symbolic calculus integrals
  - Look-up combined with numerical integration
  - Allows for integration of any function
  - Recall a function is simply one-to-on and onto
  - Means that a vertical line crosses the function only once!