CPE 381: Fundamentals of Signals and Systems for Computer Engineers

Continuous & Discrete Representation

Rahul Bhadani

Electrical & Computer Engineering, The University of Alabama in Huntsville

Outline

1. Continuous and Discrete Representations

2. Numerical Computation in MATLAB

Continuous and Discrete Representations

- **Continuous-time signals:** Depend continuously on time.
- **Discrete-time signals:** Sequences of measurements typically made at uniform times.
- **Sampling process:**

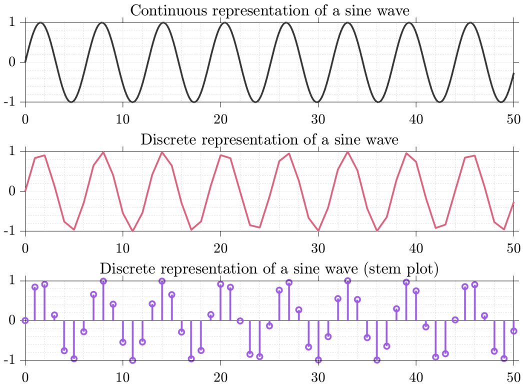
$$x[n] = x(nT_s) = x(t)|_{t=nT_s}$$

- **Example:**
  - Analog signal:

$$x(t) = 2\cos(2\pi t)$$

  - Sampled at $T_{s1} = 0.1$ s: $x_1[n] = 2\cos(2\pi n/10)$
  - Sampled at $T_{s2} = 1$ s: $x_2[n] = 2\cos(2\pi n) = 2$
- **Key point:** Choosing an appropriate sampling period T_s is crucial to preserve information.

Sampling Continuous Time Signals



$$x[n] = x(nT_s),\ T_s = \text{sample time.}$$

Code for the figure: https://github.com/rahulbhadani/CPE381_FA24/blob/master/Code/sampled_sine_wave.m

Sampling Process

- $x[n] = x(nT_s) = x(t)|_{t=nT_s}$
- This equation represents the **sampling process** where a continuous-time signal $x(t)$ is sampled at uniform intervals T_s to produce a discrete-time signal $x[n]$.
- **Key Points:**
  - n is an integer representing the sample index.
  - T_s is the sampling period.
  - The continuous-time signal $x(t)$ is sampled at $t = nT_s$.

Derivatives and Finite Differences

Continuous-Time Derivatives:

- Derivatives measure the rate of change of a function.
- Defined as the limit of the difference quotient as the interval approaches zero.
- Represented as $\frac{dy}{dt}$ for a function $y(t)$.

Finite Differences

Discrete-Time Derivatives:

- Finite differences approximate derivatives for discrete-time signals.
- Forward difference: $\Delta x[n] = x[n+1] - x[n]$.
- Backward difference: $\Delta x[n] = x[n] - x[n-1]$.

$x[n] = x(nT_s)$ when looking at continuous to discrete domain where T_s is the sampling time, n is any positive integer. We will look at this much later in detail.

Central Differences

Central Difference:

- Provides a more accurate approximation.
- Defined as $\delta x[n] = \frac{x[n+1]-x[n-1]}{2}$.
- Reduces error compared to forward and backward differences.

Relation between the Derivative and Finite Difference

The derivative and the finite-difference operators are not the same. In the limit, we have:

$$\frac{dx(t)}{dt}\Big|_{t=nT_s} = \lim_{T_s \to 0} \frac{\Delta[x(nT_s)]}{T_s}$$

Applications

Applications of Finite Differences:

- Used in numerical methods for solving differential equations.
- Essential in digital signal processing and control systems.
- Helps in approximating continuous-time derivatives in discrete systems.

Discrete Integration

Recall, the integration in the continuous domain is

$$I(t) = \int_{t_0}^{t} x(\tau) d\tau$$

then, we have

$$\frac{d}{dt} \int_{t_0}^{t} x(\tau) d\tau = x(t)$$

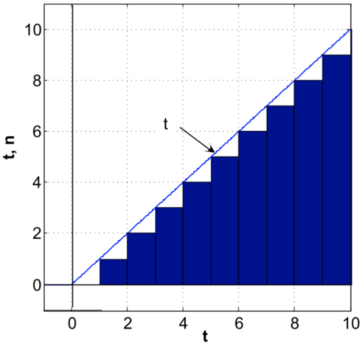
If we use D for the derivative operator, then D^{-1} is the integration operator.

$$D[D^{-1}[x(t)]] = x(t))$$

Example

- Computational integration using sums:

$$\int_0^{10} t\, dt = \frac{t^2}{2}\Bigg|_0^{10} = 50$$



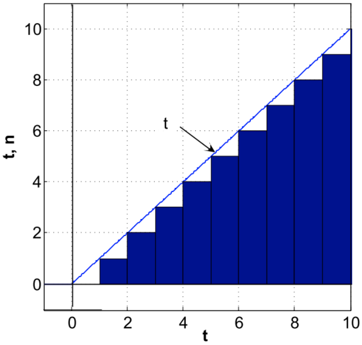
Discrete Approximation

- Approximation using pulses $(T_s = 1)$(rectangle of width $T_s = 1$, height $= n$):

$$\sum_{n=0}^{9} p[n] = \sum_{n=0}^{9} n = 45$$

- Generalized sum:

$$\sum_{n=0}^{N-1} n = \frac{N \times (N-1)}{2}$$



Improved Discrete Approximation

- Using $T_s = 10^{-3}$:

$$\sum_{n=0}^{(10/T_s)-1} nT_s^2 = \sum_{n=0}^{(10/T_s)-1} n10^{-6} == 49.995$$

The height of each pulse is nT_s and the width is T_s.

Hence, the sampling time (or its inverse – sampling frequency matters).

Solving Ordinary Differential Equations using Numerical Methods

Recognizing that derivatives can be approximated as difference equations in the discrete domain, we have some methods to solve ordinary differential equations, suitable for computer implementations.

Some common numerical methods to solve ODE are:

- Euler's Method
- Runge-Kutta Method (4th Order)

Euler's Method

- Consider the ODE: $\frac{dy}{dt} = f(t, y)$
- Initial condition: $y(t_0) = y_0$
- Euler's method formula: $y_{n+1} = y_n + hf(t_n, y_n)$
- Example: $\frac{dy}{dt} = -2y, \quad y(0) = 1$
- Using $h = 0.1$: $y_{n+1} = y_n - 0.2y_n = y_n(1 - 0.2)$

Euler's Method

Euler's method:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

Example: Solve $y' = 2t$ with $y(0) = 1$ using Euler's method with $h = 0.1$

- In this case, $f(t, y) = 2t$

t_n	y_n	y_{n+1}
0	1	1.0
0.1	1.0	1.02
0.2	1.02	1.06
$\vdots$	$\vdots$	$\vdots$

Runge-Kutta Method (4th Order)

- Consider the ODE: $\frac{dy}{dt} = f(t, y)$
- Initial condition: $y(t_0) = y_0$
- Runge-Kutta method formula:

$$\begin{aligned}
k_1 &= hf(t_n, y_n) \\
k_2 &= hf\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right) \\
k_3 &= hf\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right) \\
k_4 &= hf(t_n + h, y_n + hk_3) \\
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}$$

- Example: $\frac{dy}{dt} = t + y, \quad y(0) = 1$
- Using $h = 0.1$:
  Write a MATLAB code to implement that.

Runge-Kutta Methods

Runge-Kutta methods:

- More accurate than Euler's method
- Use multiple function evaluations at each step

Example: Solve $y' = 2x$ with $y(0) = 1$ using the Runge-Kutta method of order 4 with $h = 0.1$

x_n	y_n	y_{n+1}
$\vdots$	$\vdots$	$\vdots$

Differentiation in MATLAB

$$y(t) = \cos(t^2)$$

Differentiation in MATLAB

```
%% Solving derivatives symbolically
% y(t) = cos(t^2)
% dy/dt = -2*t*sin(t^2)
%% Symbolic derivative: the ground truth
syms t y z % we define symbols
y = cos(t^2);
z = diff(y);
figure(1);
subplot(2, 1, 1)
% symbolic plotting
fplot(y, [0, 2*pi], 'LineWidth', 3);
grid on;
hold on;
subplot(2, 1, 2);
fplot(z, [0, 2*pi], 'LineWidth', 3);
grid on;
hold on;
```

```
%% Numerical derivative
Ts = 0.1;
t1 = 0:Ts:2*pi;
y1 = cos(t1.^2);
z1 = diff(y1)./diff(t1);
figure(1);
subplot(2, 1, 1);
stem(t1, y1, 'r');
subplot(2, 1, 2);
stem(t1(1:length(y1) -1), z1, 'm' );
```

Integration in MATLAB

$$\int \frac{x^3}{x+2}dx$$

$$\int \frac{x^3}{x+2}dx = \int \frac{x^3+8-8}{x+2}dx$$ Add and subtract 8 to the numerator

$$= \int \frac{(x+2)(x^2-2x+4)-8}{x+2}dx$$ Factor the numerator

$$= \int \left(x^2-2x+4-\frac{8}{x+2}\right)dx$$ Split the fraction

$$= \int x^2dx - 2\int xdx + 4\int dx - 8\int \frac{1}{x+2}dx$$ Split the integral into separate terms

$$= \frac{x^3}{3} - x^2 + 4x - 8\ln|x+2| + C$$ Evaluate each integral

Integration in MATLAB

```
%% Symbolic Integration
% x^3/(x + 2) dx
syms x
f = x^3/(x + 2)
% indefinite integral
int(f)
% definite integral
int(f, 0, 10)
```

```
%% Numerical Integration using trapezoidal rule,
%% Numerical integration is always definite
x = 0:0.1:10;
f = x.^3./(x + 2);
trapz(x, f)
```

Ordinary Differential Equation in MATLAB

Solve the initial value problem $ty' + 3y = 0, y(1) = 2$, assuming $t > 0$. We write the equation in standard form: $y' + 3y/t = 0$.

$$P(t) = \int -\frac{3}{t}dt = -3\ln t$$

and

$$y = Ae^{-3\ln t} = At^{-3}$$

Substitute to find A: $2 = A(1)^{-3}$, so the solution is $y = 2t^{-3}$.

Ordinary Differential Equation in MATLAB

```
%% ty; + 3y = 0; y(1) = 2
% solution y = 2/t^3
syms y(t)
eqn = diff(y, t) + 3*y/t ==0;
cond = y(1) == 2;
y(t) = dsolve(eqn, cond);
```