Esai Perubahan Kode Konvolusi Diskrit

Kode awal yang dibuat berfungsi untuk memvisualisasikan proses konvolusi diskrit secara animasi. Pada implementasi tersebut, sinyal masukan x[n] dan respon impuls h[n] ditetapkan secara sederhana, masing-masing dengan panjang pendek yaitu x = [1 2 3] dan h = [1 1]. Hasil yang ditampilkan berupa tiga subplot yang menggambarkan sinyal masukan, sinyal respon impuls yang dibalik serta digeser, dan hasil perkalian titik demi titik. Dengan cara ini, kode memberikan gambaran dasar bagaimana proses konvolusi berlangsung. Namun, kode tersebut masih memiliki keterbatasan dalam hal fleksibilitas dan informasi hasil akhir.

Setelah dilakukan perubahan, kode menjadi lebih **generalis**. Pertama, sinyal masukan x dan respon impuls h dapat diubah menjadi panjang berapapun, tidak terbatas pada contoh sederhana seperti versi awal. Hal ini membuat kode bisa digunakan untuk berbagai macam eksperimen sinyal, baik yang pendek maupun panjang. Kedua, hasil konvolusi kini disimpan dalam sebuah array y sehingga dapat ditampilkan secara bertahap. Perubahan ini ditunjukkan melalui penambahan subplot keempat yang memperlihatkan perkembangan hasil konvolusi lengkap dari awal hingga indeks ke-n. Dengan demikian, selain animasi tiap langkah, pengguna juga bisa mengamati bagaimana bentuk akhir sinyal konvolusi terbentuk secara kumulatif.

Perbedaan utama sebelum dan sesudah perubahan terletak pada **cakupan fungsionalitas** dan **kedalaman visualisasi**. Kode awal hanya memberikan gambaran proses pergeseran dan perkalian titik demi titik, sedangkan kode hasil modifikasi mampu menampilkan hasil konvolusi lengkap yang terus diperbarui selama animasi berlangsung. Selain itu, kode awal hanya sesuai untuk contoh kecil, sementara kode baru bisa diaplikasikan pada kasus yang lebih umum tanpa harus dimodifikasi secara manual. Dengan pengembangan ini, kode menjadi lebih bermanfaat tidak hanya untuk memahami konsep dasar konvolusi, tetapi juga untuk keperluan praktikum atau demonstrasi yang lebih kompleks.

Perbedaan Penggunaan trapz dan integral dalam Perhitungan Integral di MATLAB

Dalam komputasi numerik, khususnya ketika menghitung integral pada MATLAB, terdapat beberapa metode yang dapat digunakan. Dua di antaranya adalah fungsi **trapz** dan **integral**. Keduanya sama-sama digunakan untuk menghitung luas di bawah kurva, tetapi memiliki karakteristik, pendekatan, dan kelebihan masing-masing.

Fungsi **trapz** menggunakan metode **trapezoidal rule**, yaitu pendekatan numerik dengan membagi kurva menjadi potongan-potongan trapesium, kemudian menjumlahkan luasnya. Agar dapat digunakan, trapz memerlukan data dalam bentuk **vektor diskrit** berupa pasangan (t, f(t)). Oleh karena itu, akurasi perhitungan trapz sangat bergantung pada kepadatan titik sampel. Semakin rapat grid waktu yang digunakan, hasil integral semakin mendekati nilai sebenarnya. Namun, apabila jumlah titik terlalu sedikit atau fungsi memiliki perubahan yang tajam, trapz dapat menghasilkan error yang cukup besar. Keunggulan utama trapz adalah kesederhanaan dan kecepatan eksekusi, sehingga sering dipakai dalam animasi atau simulasi real-time.

Di sisi lain, **integral** pada MATLAB menggunakan pendekatan **adaptive quadrature**. Fungsi ini bekerja dengan cara mengevaluasi integral dari sebuah **function handle** (@(t)) dalam interval tertentu. Berbeda dengan trapz yang membutuhkan data diskrit, integral menghitung langsung dari bentuk fungsi, kemudian secara otomatis menyesuaikan titik-titik evaluasi agar hasilnya lebih presisi. Keunggulan integral adalah akurasinya yang tinggi meskipun interval besar atau jumlah sampel sedikit. Namun, integral membutuhkan definisi fungsi yang jelas dan relatif lebih lambat daripada trapz bila dipanggil berulang kali dalam loop animasi.

Dengan demikian, perbedaan utama antara keduanya terletak pada **representasi data dan tingkat akurasi**. trapz cocok digunakan ketika data hanya tersedia dalam bentuk diskrit (misalnya hasil pengukuran eksperimen) atau ketika kecepatan lebih diprioritaskan daripada ketelitian. Sebaliknya, integral lebih unggul dalam presisi matematis, terutama jika fungsi dapat didefinisikan secara eksplisit dalam bentuk analitik.

Secara praktis, trapz lebih sering dipakai pada konteks simulasi cepat dan pengolahan data hasil eksperimen, sedangkan integral lebih sesuai untuk validasi perhitungan teoritis atau saat ketelitian numerik menjadi hal yang utama. Oleh karena itu, pemilihan metode sebaiknya disesuaikan dengan kebutuhan: **efisiensi komputasi** atau **akurasi hasil**.