Universidade Federal do Rio de Janeiro
Departamento de Ciência da Computação
Instituto de Matemática

Cálculo Numérico

Animação com Diferenças Centradas (Trabalho Final)

Aluno:
Filipe José Maciel Ramalho *(DRE: 114032785)*
Professor:
João Paixão

Sumário

Introdução

Atualmente, com o grande interesse por jogos e animações cada vez mais realistas, desenvolvem-se técnicas cada vez mais elaboradas para aproximar o artificial do real.

A disciplina de Cálculo Numérico tem como principal finalidade justamente a aproximação do “artificial” com o real. Através dos métodos numéricos é possível conseguir valores discretos muito próximos da realidade de funções matemáticas, inclusive das que não possuem nenhuma solução analítica possível de ser calculada. Porém alguns desses métodos, mesmo muito aproximados, se mostram bem ineficientes dependendo do grau de precisão que a aplicação necessita.

Este trabalho está baseado no funcionamento do método de diferenças centradas. Este método é caracterizado por calcular de forma discreta a derivada aproximada de uma função. Fazendo manipulações algébricas é possível descobrir a relação de recorrência que representa os valores de uma determinada função a uma certa taxa no domínio, e é exatamente nesta relação de recorrência que se baseia o projeto.

Metodologia

Inicialmente o projeto visava a construção de um jogo baseado em mecânica clássica e força elétrica entre partículas, mas devido a problemas que acarretaram na demora no cumprimento do prazo e restrições de complexidade de processamento foi necessário a redução drástica do seu escopo, contemplando-se assim somente as interações de colisão entre “paredes” e “esferas” e abrangendo apenas o movimento de queda livre com resistência de fluidos.

A fórmula foi desenvolvida baseada no estudo das equações de movimento envolvidas no projeto: a equação da força de resistência do ar (F_{res} = Kv, com constante de resistência do fluido (k) negativa) e a fórmula de movimento em queda livre baseada na força peso (P = mg). A fórmula final também foi desenvolvida baseada na segunda Lei de Newton:

$$\sum_{i=1}^{n} F_i = ma$$

Desta forma,o desenvolvimento da equação com essas fórmulas combinadas fica:

$$mg + kv = ma$$

segundo as diferenças finitas centradas,

$$v = \frac{(s(t-h)-s(t+h))}{2h}$$

$$a = \frac{(s(t-h)-2s(t)+s(t+h))}{h^2}$$

Substituindo “v” e “a” na fórmula da força:

$$mg + \frac{k(s(t-h)-s(t+h))}{2h} = \frac{m(s(t-h)-2s(t)+s(t+h))}{h^2}$$

$$mg + \frac{ks(t-h)-ks(t+h)}{2h} - \frac{ms(t-h)-2ms(t)+ms(t+h)}{h^2} = 0$$

$$mg + \frac{khs(t-h)-khs(t+h)-2ms(t-h)+4ms(t)-2ms(t+h)}{2h^2} = 0$$

$$2h^2mg + khs(t-h) - khs(t+h) - 2ms(t-h) + 4ms(t) - 2ms(t+h) = 0$$
$$2h^2mg + s(t-h)[kh-2m] + 4ms(t) = s(t+h)[kh+2m]$$

$$\frac{2h^2mg+s(t-h)[kh-2m]+4ms(t)}{[kh+2m]} = s(t+h)$$

Como a relação entre o passo atual e o seguinte é linear, podemos reescrever a equação como segue:

$$S_{(p+2)} = \frac{2h^2mg+S_{(p)}(kh-2m)+4mS_{(p+1)}}{kh+2m}$$,onde p é o número do passo.

Considerando o plano (x,y), podemos escrever esta equação da seguinte forma (sabendo que g é atuante apenas no eixo y).

$$S_{y(p+2)} = \frac{2h^2mg+S_{y(p)}(kh-2m)+4mS_{y(p+1)}}{kh+2m}$$ e

$$S_{x(p+2)} = \frac{S_{x(p)}(kh-2m)+4mS_{x(p+1)}}{kh+2m}$$ (aqui g é 0),

Onde:
- S_x = *Posição em x*
- S_y = *Posição em y*
- p = *Número do passo*
- h = *Valor do intervalo entre passos no domínio*
- m = *Massa do corpo*
- g = *Gravidade*
- k = *Constante de resistência do fluido*

Como é possível ver, uma determinada posição depende do valor das duas posições anteriores a ela, desta forma se faz necessário duas posições iniciais para inicializar a relação de recorrência, o que faz sentido considerando que uma equação diferencial de segunda ordem necessita de duas condições iniciais para ser resolvida.

Como esta relação de recorrência funciona com base em passos discretizados de uma função contínua, existe sempre um erro associado a estes valores calculados. Este erro, considerando apenas as diferenças finitas , é calculado como se segue:

Como o cálculo do erro em diferenças centradas de primeira ordem já foi definido durante a aula, tomei a liberdade de não escrever este cálculo aqui, mas apenas sua forma final.

- para $a-h \leq t \leq a+h$ e $M_3 = max(f'''(t))$

$$E = \frac{M_3 h^2}{3!}$$ (definido em sala de aula)

Para diferenças centradas de segunda ordem será desenvolvido o seguinte cálculo:

- para $a-h \leq t \leq a+h$ e $M_4 = max(f^{(4)}(t))$
- Usando Polinômio de Taylor

$$f(a-h) = f(a) - f'(a)h + \frac{f''(a)h^2}{2} - \frac{f'''(a)h^3}{3!} + \frac{f^{(4)}(\varepsilon_1)h^4}{4!}$$
$$f(a+h) = f(a) + f'(a)h + \frac{f''(a)h^2}{2} + \frac{f'''(a)h^3}{3!} + \frac{f^{(4)}(\varepsilon_2)h^4}{4!}$$

Somando os lados...

$$f(a-h)+f(a+h) = f(a) - f'(a)h + \frac{f''(a)h^2}{2} - \frac{f'''(a)h^3}{3!} + \frac{f^{(4)}(\varepsilon_1)h^4}{4!} +$$
$$f(a) + f'(a)h + \frac{f''(a)h^2}{2} + \frac{f'''(a)h^3}{3!} + \frac{f^{(4)}(\varepsilon_2)h^4}{4!}$$

Manipulando, obtemos a seguinte equação:

$$\frac{f(a-h)-2f(a)+f(a+h)}{h^2} - f''(a) = \frac{h^4}{4!h^2}(f^{(4)}(\varepsilon_1) + f^{(4)}(\varepsilon_2))$$

$$\frac{f(a-h)-2f(a)+f(a+h)}{h^2} - f''(a) = \frac{2M_4 h^2}{4!}$$

$$E = \frac{M_4 h^2}{12}$$

Somando os erros teremos:

$$E_{total} = \frac{M_{v3} h^2}{6} + \frac{M_{a4} h^2}{12} = \frac{(M_4+2M_3)h^2}{12}$$

que é bem eficiente, uma vez que a taxa em que ele é reduzido é da ordem de $O(h^2)$

Resultados

A estrutura de diretório do projeto foi construída de forma atômica, como uma casca de cebola, onde os arquivos mais fundamentais ficam mais no interior e os arquivos que dependem dos arquivos fundamentais ficam mais externamente.

Como o projeto foi construído usando html, css e js, foi necessário a criação do arquivo loadfiles, que é o responsável por carregar quase todos os objetos necessários ao funcionamento do projeto. Através das funções de importação, ele adiciona as urls dos arquivos dos objetos no html. Desta forma, sempre que um objeto depender de um outro objeto em outro arquivo, basta importar tal arquivo usando as funções de import.

O projeto funciona usando o canvas do HTML5 e suas respectivas funções em javascript. O sistema usa para o cálculo das diferenças finitas uma função de espera chamada setInterval que executa em loop o mesmo código um intervalo de tempo determinado, mas tal código é limitado a ao tempo mínimo que ele consegue reproduzir (que no caso é de 1 milissegundo), porém para fins de segurança o programa não executa os cálculos da diferença finita a um intervalo inferior a 5ms. Para garantir que mesmo alterando a precisão do cálculo de diferença finita não altere o tempo final de percurso de um ponto a para um ponto b, é necessário que ele processe as diferenças finitas no mesmo intervalo h estipulado, então foi criado uma nova variável n que define quantos “tempos” e h’s por segundo o cálculo terá. Logo o cálculo de h’s e tempo de processamento em função de n será:

$$h = \frac{1}{n}$$
$$t = \frac{1000}{n}$$

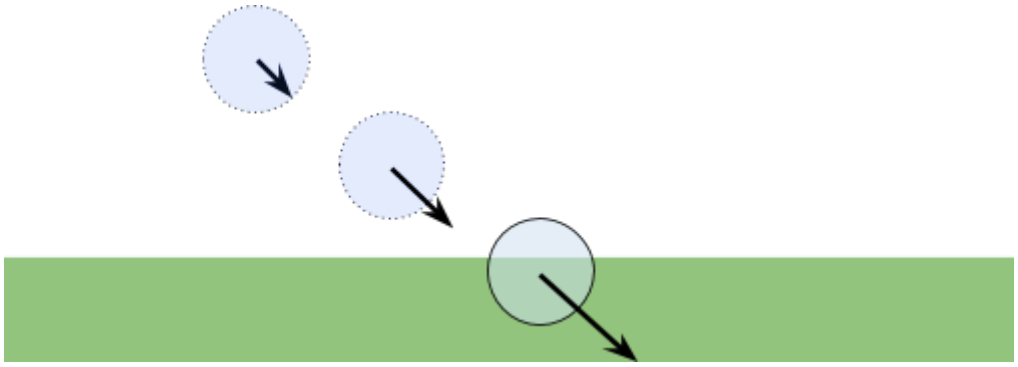
Além disso foram adicionadas duas variáveis modificadoras para alterar h ou n independentemente um do outro para que o tempo seja “encurtado” e a velocidade de reprodução seja alterada por um multiplicador (como na velocidade de reprodução de um vídeo). Com isso, “h” e “t” ficam calculados da seguinte forma:

$$h = \frac{1}{n * xspeed1}$$
$$t = \frac{1000}{n * xspeed2}$$

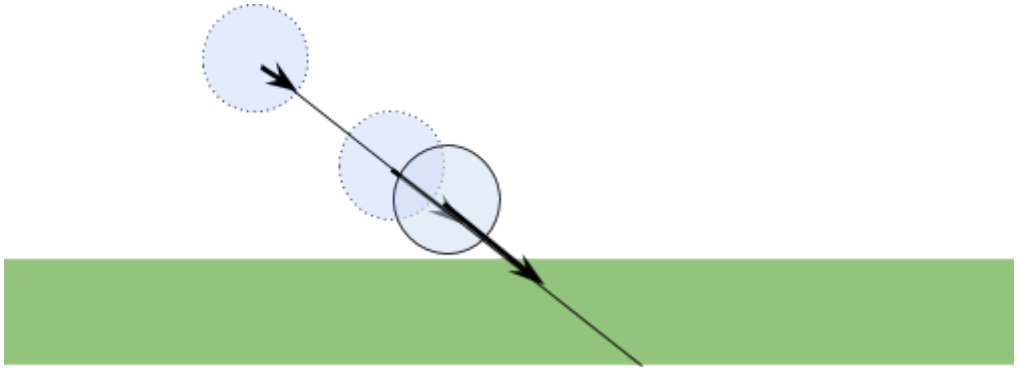
Um grande problema do projeto foi o fato ser necessário trabalhar com funções não deriváveis em determinados pontos. Tais pontos são justamente os pontos de colisão dos corpos no programa.

Outro fato também ruim a respeito das colisões é o processo de detecção, o qual é processado pelo algoritmo “SAT”(Separated axis theorem) que sozinho tem complexidade $o(n^2)$ para cada par de elementos dentro da zona de detecção delimitada pelas “bounding boxes”.

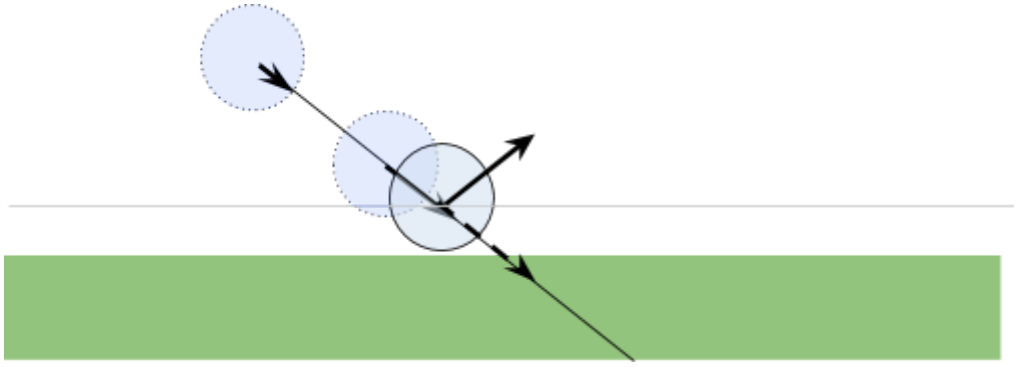
Assim que o objeto detecta a colisão com uma “parede”, o código executa as ações conforme desenhado a seguir. Para o correto funcionamento do algoritmo SAT é necessário que um objeto “entre” dentro do outro, por isso é necessário, quando identificada a colisão, deslocá-lo como que “voltando no tempo” para a posição imediatamente antes de entrar no outro corpo.



Detecta colisão e pára as diferenças finitas;



Volta no tempo para a posição imediatamente antes de colidir.



Reflete o vetor velocidade e reinicia o loop das diferenças finitas, com as novas posições calculadas através do vetor velocidade;

O grande problema é que durante a colisão existe um erro muito grande quando faz-se a “volta no tempo” pois o vetor velocidade deveria também ser alterado, mas alterar a velocidade é muito difícil pois geralmente a posição antes de colidir não é um valor discretizado pelas diferenças finitas, mas sim um valor entre dois valores discretizados. Então foi necessário criar uma função de correção baseada na equação de Torricelli, porém tal função (que é o Bháskara de uma função do segundo grau) pode fornecer números complexos e/ou inconsistências com divisão por zero. Então tentei com o auxílio do site Desmos achar uma função que melhor descrevesse esta nova velocidade, porém esta função também não fornece o resultado correto (apesar de conseguir diminuir um pouco este erro).

Finalmente, para diminuir o erro de forma satisfatória, a solução encontrada foi remover uma colisão perfeitamente elástica e atribuir um coeficiente de restituição diferente de 1.

Conclusão

Os resultados e os cálculos referentes ao projeto foram bastante satisfatórios com relação às experiências adquiridas e observações feitas a respeito do quão complicado pode ser criar uma animação, por mais simples que ela possa parecer.

Uma observação bastante interessante é justamente o fato de a altura máxima ir diminuindo de forma logarítmica a cada pulo, o que sugere que segundo às leis da Física o corpo nunca pararia de pular a não ser que ele pulasse num momento infinito, mas claramente não é isto que ocorre no mundo físico. A conclusão mais provável para esse fato é que o computador, assim como no método de Newton (que é preciso no infinito), não consegue processar de forma contínua estes pulos pois não possui memória infinita e por isso quando a altura máxima chega perto de sua última casa decimal ele a arredonda para cima.

Outra conclusão até bastante óbvia com relação a este projeto é sobre a dificuldade de realizar diferenças finitas em qualquer função que não possui todos os pontos deriváveis, pois dividir tal função em duas partes para então executar diferenças finitas em cada uma mostra-se bastante ineficiente e impreciso.

- **Links do projeto no gitHub**
  - **Página:** **https://filipermlh.github.io/TFCalcNum/**
  - **Repositório:****https://github.com/filipeRmlh/TFCalcNum**

Referências

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference
http://jriecken.github.io/sat-js/docs/SAT.html
http://www.portalsaofrancisco.com.br/fisica/resistencia-do-ar
http://www.dyn4j.org/2010/01/sat/