CAPÍTULO 4

IMPLEMENTAÇÃO COMPUTACIONAL NA SIMULAÇÃO DE CORRIDAS DE DETRITOS

4.1. Estruturação básica do programa SAND

O programa SAND foi desenvolvido para aplicação a problemas relacionados à produção de areia em poços de petróleo [3] considerando fluxo mono e bifásico. Na verdade, SAND é a extensão gráfica de uma biblioteca programada na linguagem C++ que analisa os dados de entrada ao programa e faz os cálculos respectivos.

Usando a programação orientada a objetos (POO), Campos et al (2002) [3] desenvolveram uma biblioteca (DEMlib) com a idéia de implementar o DEM e que de vez facilitasse a manutenção e extensão da mesma para sua aplicação a outros problemas relacionados com este método. Usando a definição de objetos, elementos e classes [52], [53] conseguem-se atingir as mais diversas condições e geometrias dos problemas a serem tratados com este método numérico. Assim, a biblioteca DEMlib consiste em um conjunto de classes que definem objetos básicos que são usados na implementação do DEM e uma extensão gráfica como já se mencionou.

SAND foi programado nas linguagens C e C++, com o *toolkit* de interface IUP/LED e o sistema gráfico CD, proporcionando as seguintes facilidades:

- Interface gráfica para visualização dos passos de análise.
- Definição interativa das paredes que definem as condições de contorno do problema e dos discos que representam o meio granular.
- Visualização das forças de contato e atuantes no centróide de cada partícula.
- Interface com a linguagem de programação interpretada LUA, que permite a geração de discos com dimensões definidas segundo as regras definidas pelo usuário.

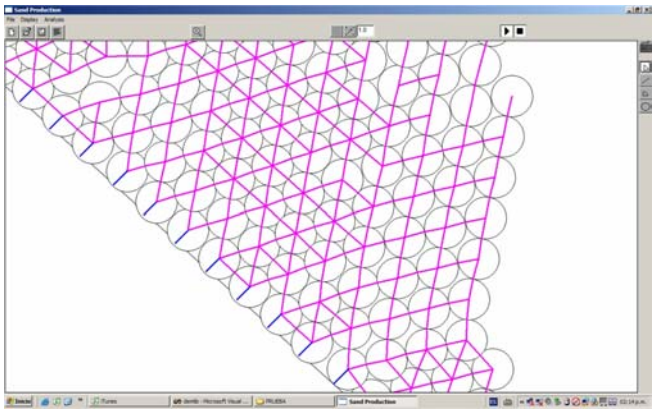


FIGURA 4.1- Interface gráfica do Programa SAND mostrando os contatos entre elementos.

De forma geral, pode-se falar que o programa SAND segue a metodologia descrita no capítulo anterior para a implementação do DEM. Uma forma simplificada e seguida no desenvolvimento do mesmo programa se mostra a continuação.

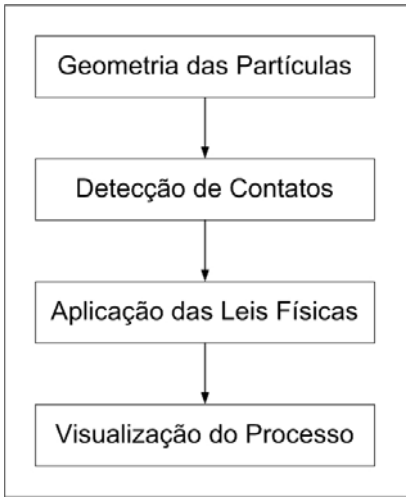


FIGURA 4.2- Etapas de implementação computacional do DEM.

4.1.1. Geometria das Partículas.

Nesta etapa consideram-se duas etapas. A primeira relacionada com a geração da geometria dos elementos discretos e a determinação das propriedades geométricas. A segunda referida à geometria dos anteparos (paredes) de escoamento ou condições de contorno.

A geração das partículas é feita com uma rotina simples que permite a geração de uma malha regular ou densa. Aliás, permite-se gerar uma malha com diâmetro e densidade variáveis (Vede FIGURA 4.3). O arquivo de saída está formatado para o programa de interface LUA 5.0 que é utilizado pelo SAND para ler ditas propriedades.

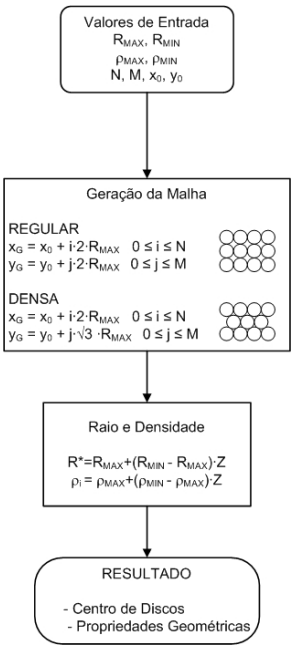


FIGURA 4.3- Esquema para a geração de elementos discretos.

De forma geral, o programa armazena estas informações no objeto BLOCK que representa um elemento discreto genérico (Vede FIGURA 4.4). As principais características contidas neste objeto são: área, volume, inércia, coordenadas do centróide, velocidade e deslocamentos do elemento. Métodos particulares de cálculo de algumas propriedades segundo o tipo de partícula são contidas nas classes PBlock e DBlock.

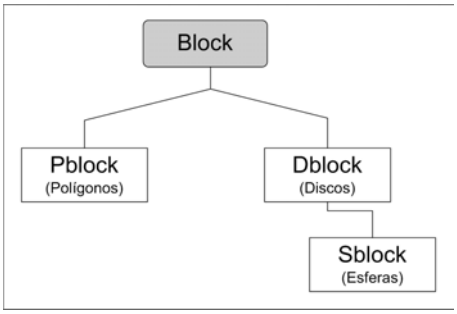


FIGURA 4.4- Hierarquia de classes para a definição geométrica do elemento discreto.

Por outro lado, estabelecem-se as condições do contorno do problema definindo os anteparos dos elementos discretos. Num início, SAND foi desenvolvido para o caso de poucos anteparos confinantes, geralmente em forma de caixote. Os primeiros elementos criados foram segmentos retos, polígonos e arcos de círculo. Estas classes estão contidas no objeto WALL (Vede FIGURA 4.5).

Este trabalho incorpora uma nova sub-rotina para representar uma função curva bidimensional mediante o algoritmo de interpolação *curva spline cúbica*. Foi programada a sub-rotina SPLINE que segmenta a função cúbica, considerando a definição de curvatura (κ_{curv}) admissível em cada ponto ($\kappa_{curv} \leq 0.25$), em segmentos lineares de menor tamanho conseguindo uma representação curva de um elemento de parede. O arquivo de saída é formatado para ser usado pela classe LWall.

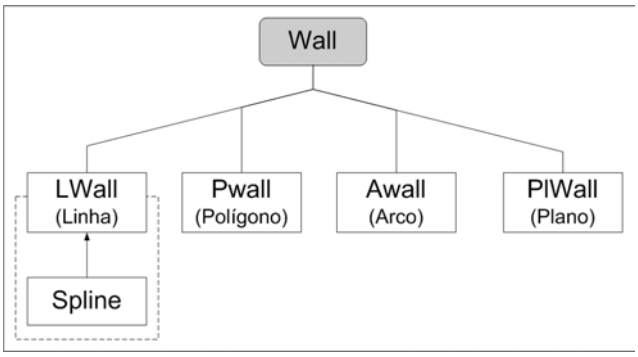


FIGURA 4.5- Hierarquia de classes para a definição dos tipos de anteparos no programa SAND.

O uso das Spline cúbicas se justifica respeito a outras ordens, pois os pontos de inflexão para maiores ordens não fazem diferença visual apreciável e estes não se apresentam em ordens menores pelo que dificulta a continuidade [54]. Em geral, uma curva spline cúbica $S_n^3(x)$ tem as seguintes características [55]:

- É um polinômio cúbico para cada par sucessivo de pontos do conjunto total de pontos dados n.
- A primeira e segunda derivada desta função são continuas no intervalo de estudo.
- A tangente final em cada intervalo é igual à tangente inicial do intervalo seguinte.

- É preciso conhecer uma condição inicial para poder resolver o sistema de incógnitas (coeficientes do polinômio cúbico). Se a segunda derivada do ponto inicial ou final do conjunto for conhecida e nula, diz-se *condição de contorno natural*. Se a tangente do ponto inicial ou final do conjunto for conhecida e não nula, diz-se de *condição de contorno fixa.*

O algoritmo usado se mostra de forma esquemática na FIGURA 4.6.a e FIGURA 4.6.b. Maiores detalhes sobre este algoritmo são apresentados no Anexo V.

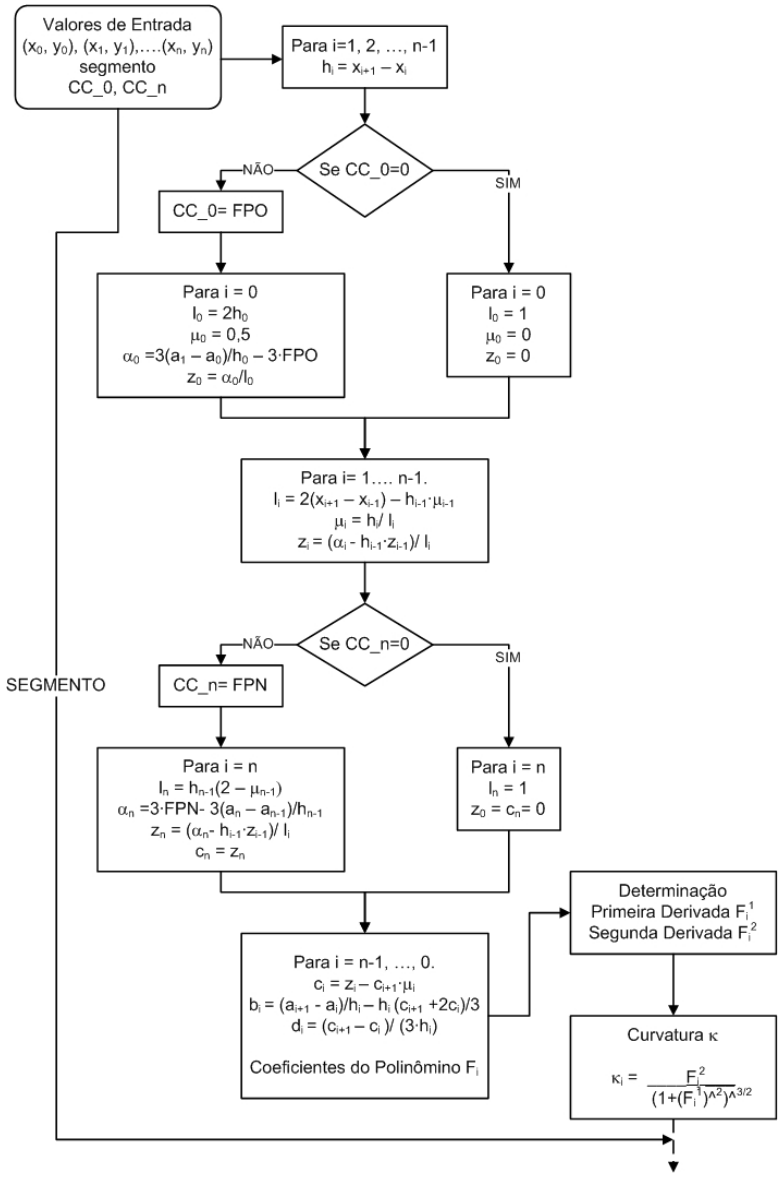


FIGURA 4.6.a- Esquema da rotina SPLINE.

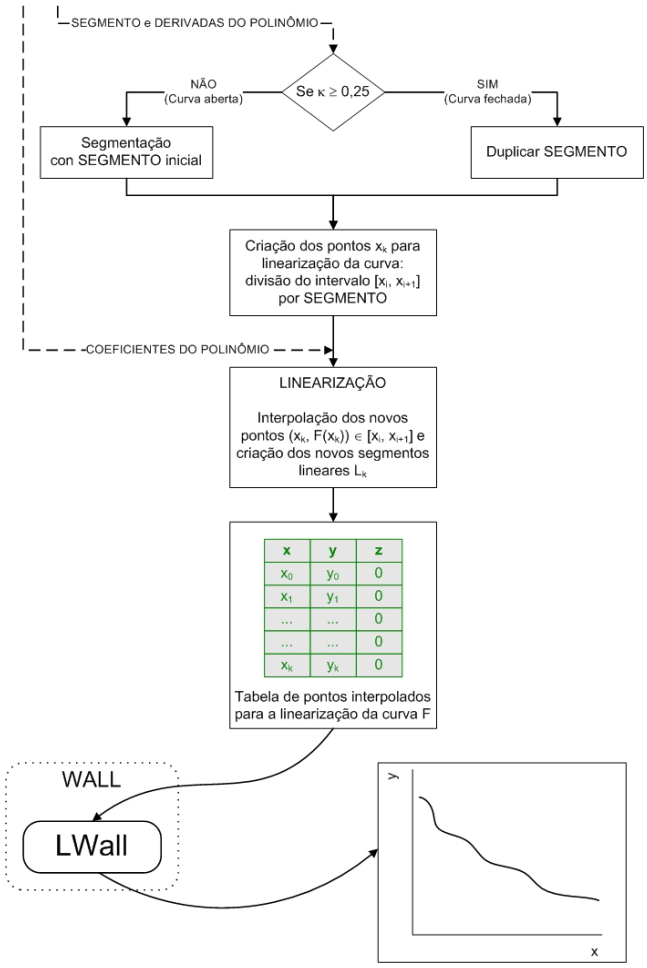
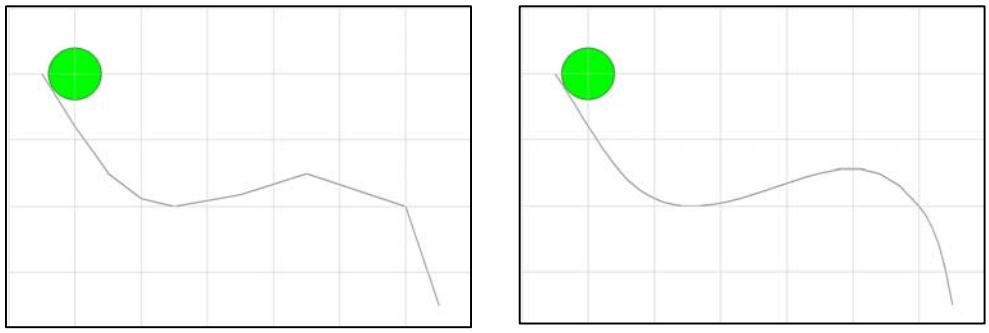


FIGURA 4.6.b- Esquema da rotina SPLINE (continuação).

Desta forma, representa-se qualquer tipo de curva mediante certo número de segmentos lineares de tamanho variável segundo a curvatura do ponto em estudo. Uma saída típica do programa SAND do algoritmo anterior se mostra na seguinte figura.



(a) (b)

FIGURA 4.7- Representação gráfica de anteparo (a) linear (b) Spline cúbico linearizado com 5 segmentos.

Existem alguns outros algoritmos de curvas spline até de quinta ordem propriamente dito, ainda que a teoria permita ordens maiores estas são referidas a outros algoritmos mais eficientes como polinômios de Hermite, polinômios de Bézier, polinômios de Berstein ou NURBS [54]. Estes métodos são recomendados para interpolações de curvas ou superfícies tridimensionais que representam o terreno.

4.1.2. Detecção de Contatos.

Uma vez conhecida a geometria do problema é preciso avaliar a potencial existência de contato entre elementos e calcular dito ponto poder aplicar as leis físicas respectivas. Esta é a etapa crítica do DEM, pois o algoritmo utilizado para a detecção determina o tempo de cálculo e a capacidade de memória dinâmica consumida. Por exemplo, para um arranjo de N partículas, o número de iterações uma a uma a serem feitas para conferir a existência ou não de contato é de N(N-1)/2, pelo qual é preciso algoritmos mais eficientes [24]. Aliás, o problema complica-se ainda mais para geometrias não-circulares de partículas e paredes de forma irregular em grande número.

Este problema tem sido objeto de estudo em aplicações de modelagem geométrica, gráfica computacional e robótica [56]. Este processo usualmente é feito em duas etapas. A primeira identifica o par de objetos que poderiam potencialmente estar em contato em um determinado passo de tempo. As técnicas mais usadas têm sido a da cela adjacente (consome muita memória) e da partícula mais próxima (para arranjos grandes não é recomendado) as que são chamas de técnicas binárias pelo tipo de estrutura encadeada de dados com que são programadas (Vede ANEXO VI). Recentemente, a triangulação dinâmica tem mostrado alguns resultados muito satisfatórios, mas é de uma alta complexidade programática [24], [56]. Para este efeito, SAND originalmente usando a técnica da cela adjacente subdivide o domínio total do problema geométrico em unidades menores (box) para reduzir o tempo de cálculo e o aumento da memória armazenada na busca potencial de contactos entre partículas muito distantes. Estes subdomínios são representados por caixas bidimensionais com dimensões estabelecidas pelo usuário, mas como mínimo devem ser maiores o iguais ao menor diâmetro existente de partícula. Caixas muito grandes ou muito pequenas não são escolhas eficazes para reduzir o tempo de cálculo [56]. Desta forma, cada

partícula é referida a cada subdomínio com uma lista contendo também a localização dos seus potenciais vizinhos. Na segunda etapa se verifica se o contato entre ditas partículas existe usando o critério da equação (3.17). Logo depois se procede a determinar o ponto de contato para efeitos de determinar as forças atuantes que dependem da magnitude da interposição entres partículas.

A partir desta subdivisão o programa determina o tipo de contato, seja este entre partícula-partícula ou partícula-anteparo como se mostra na FIGURA 4.8. Antes de determinar o tipo de contato deve-se conferir a existência real do mesmo. Este processo é dinâmico e demorado pelo que o consumo de memória é alto.

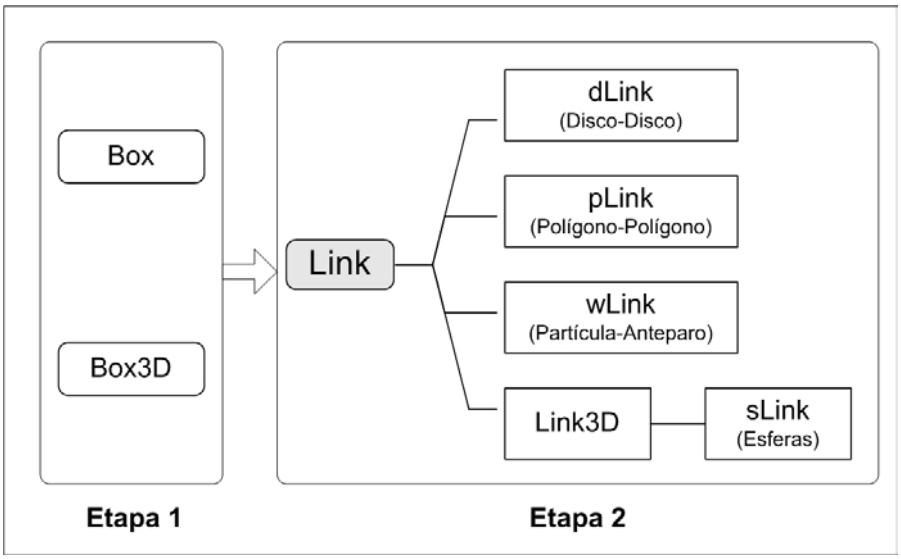


FIGURA 4.8- Etapas na detecção de contatos seguindo as hierarquias de objetos usadas no SAND.

Uma coisa importante de destacar é o fato de que o tempo de CPU na procura de contatos é independente da distribuição espacial das partículas ao igual que o armazenamento de memória [57].

Neste trabalho é usado o algoritmo de Munjiza de procura não-binária (NBS) [57]. Este método se baseia num mapeamento das partículas contidas numa janela por colunas ou celas verticais (dimensão n_y) cujas entradas contêm a primeira partícula contida em cada cela. Logo, é feito um mapeamento das partículas em celas horizontais (dimensão n_x). A detecção dos contatos a partir da cela (i,j) realiza-se nas celas adjacentes (i-1, j), (i-1, j-1), (i, j-1) e (i+1, j-1) e não em todas as celas adjacentes

como se faz nos métodos tradicionais. Desta forma o tempo diminui de $\Phi(N^2)$ a $\Phi(N\cdot(R_{max}/R_{min})^2)$ [57].

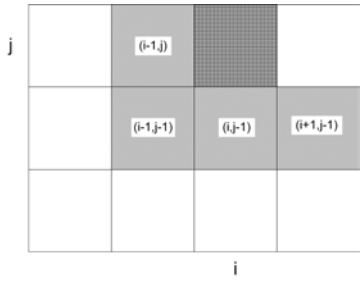


FIGURA 4.9- Sistema de celas usado na busca de contatos segundo o algoritmo Mujinza.

Para determinar os possíveis contatos numa cela procuram-se a partir do elemento j da cela n_y as partículas contidas na cela n_x. Nesta última cada partícula está referida à seguinte partícula (por isto é chamada não-binário) contida na cela onde é realizada a busca de possíveis contatos. Uma vez achadas as possíveis partículas candidatas a estarem em contato se verifica a condição da equação (3.17) para descartar contato provável de contato real. O tamanho das celas (n_x, n_y) deve ser como mínimo $\sqrt{3}\ R_{min}$ para garantir ao menos uma partícula em cada cela. A janela (searching window) deve ter um tamanho mínimo de $2\cdot R_{max}$.

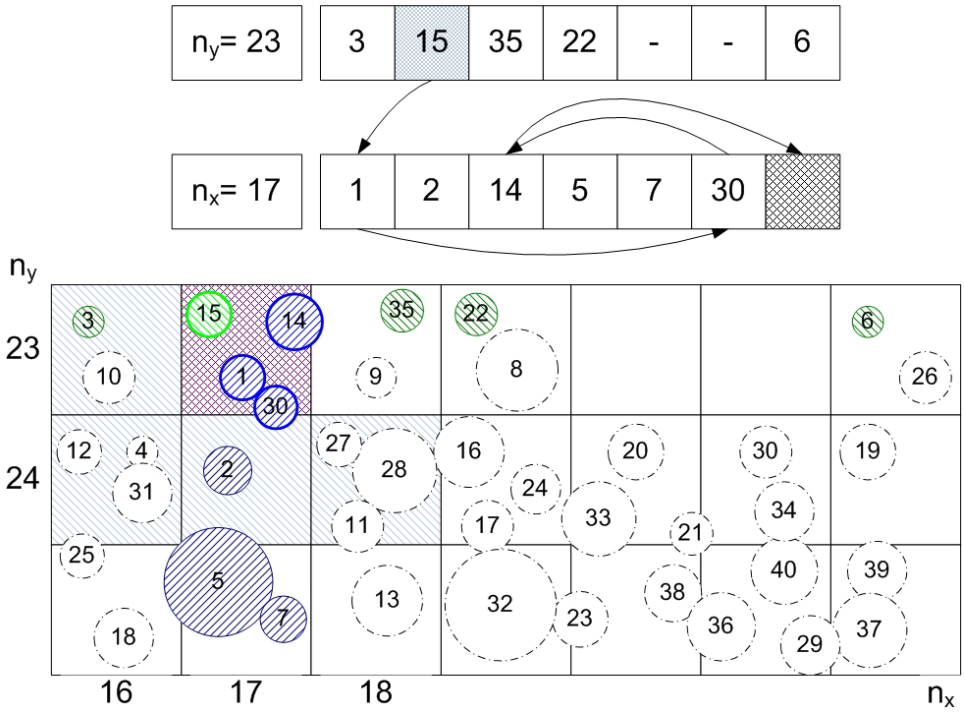


FIGURA 4.10- Exemplo de detecção de contatos na cela (i,j) segundo o algoritmo de Mujinza.

De forma geral, o ciclo do algoritmo de Mujinza é descrito como segue:

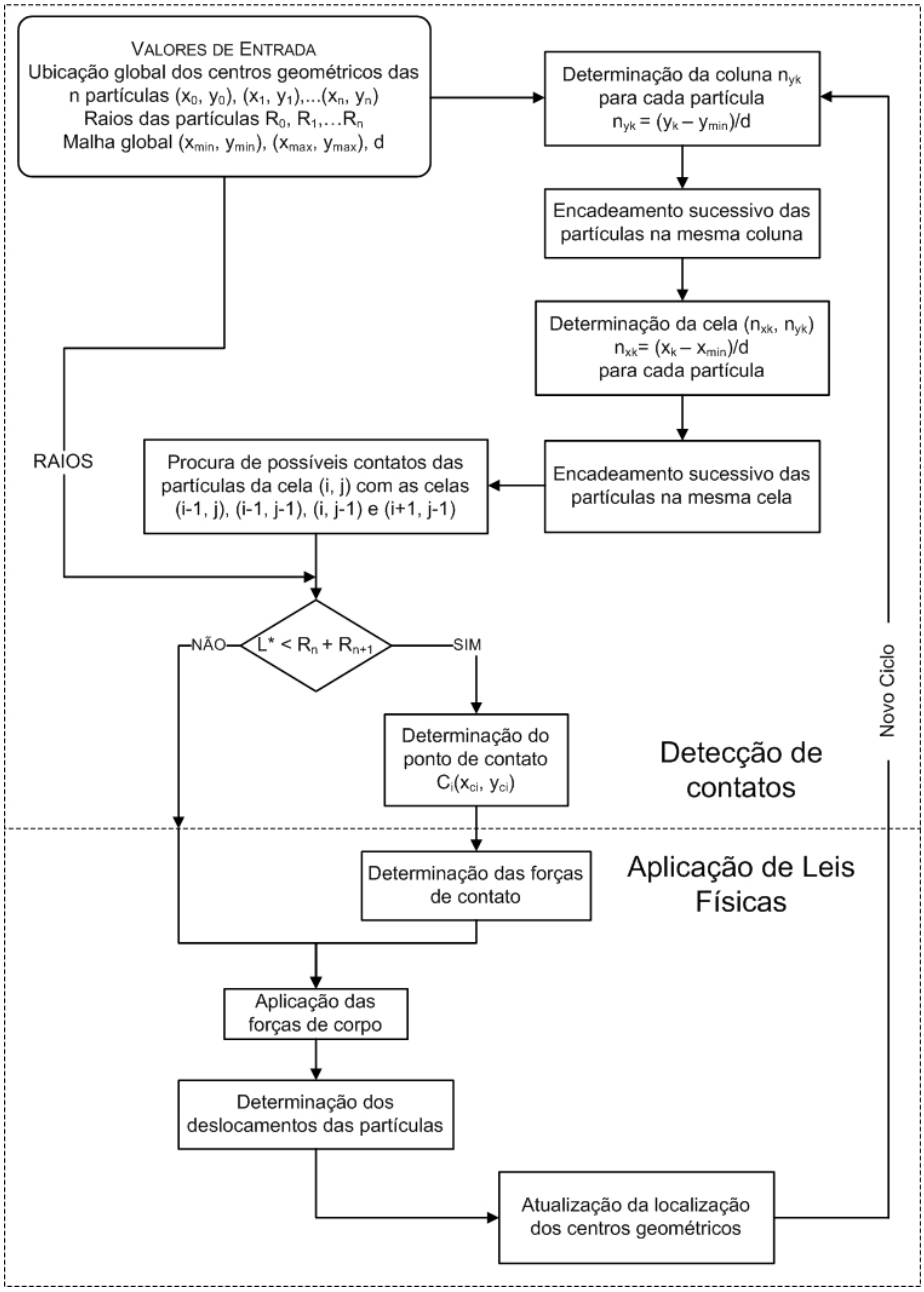


FIGURA 4.11- Ciclo de cálculo para determinar velocidades e deslocamentos das partículas a partir da detecção de contatos seguindo a algoritmo Mujinza e sua relação com a etapa de aplicação das leis físicas.

Por outro lado, para incluir os paramentos lineares dentro deste algoritmo devem-se considerar os pontos extremos que definem dita reta como uma entidade só. O problema é que uma linha contida na janela de procura estará contida

simultaneamente em várias celas. Propõe-se aqui um algoritmo de mapeamento de paramentos com igual formato do que o usado para mapear as partículas. Mas neste caso, os pontos iniciais do segmento linear definem os vértices opostos de uma área retangular que será identificada como a linha dentro da janela de procura de contatos. Deste jeito a procura de contato partícula-paramento se limita para aqueles paramentos contidos dentro da janela de procura e não se faz procura entre todos os paramentos como estava a rotina original, reduzindo o tempo de execução. Para compreender melhor o algoritmo, vede a seguinte figura.

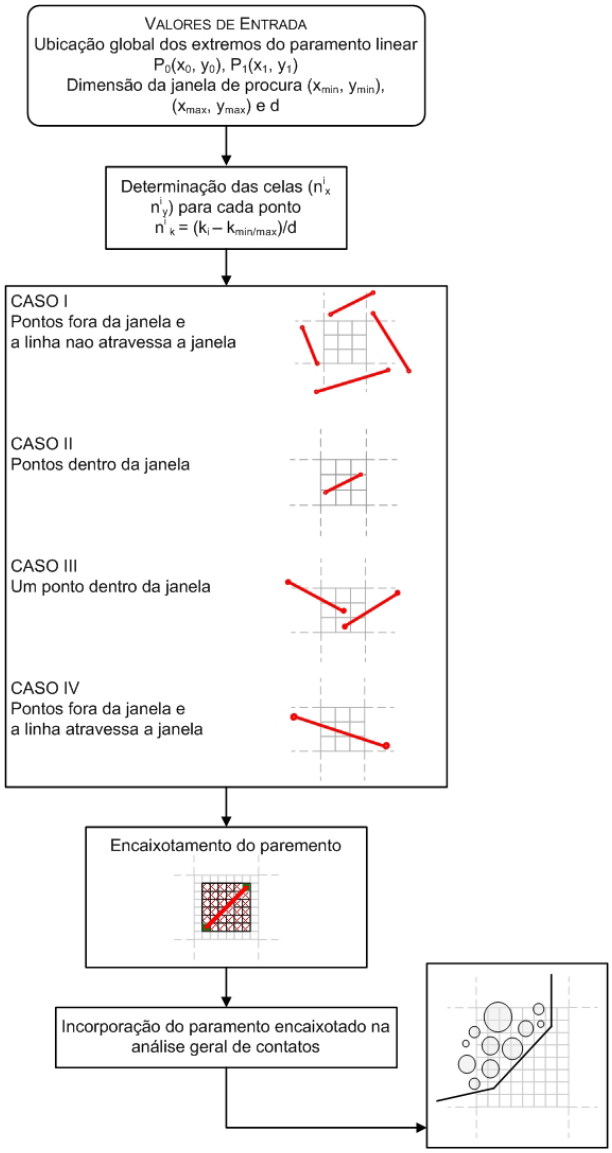


FIGURA 4.12- Esquema de programação do mapeamento dos paramentos para seu uso no algoritmo de Munjiza .

4.1.3.
Aplicação das Leis Físicas

Uma vez detectado um contato e sua localização espacial, aplica-se as formulações vistas na secção 3.2 seguindo o Modelo de Cundall para obter as forças de contato. Estas forças, junto com as forças de corpo são aplicadas nos centros de massa de cada partícula para serem incluídas nas equações do movimento geral do sistema. Assim, determinam-se as velocidades e os deslocamentos produzidos por estas forças o que leva a uma atualização na posição da partícula. Desta forma inicia-se o ciclo de cálculo para o seguinte passo de tempo como se ilustra na FIGURA 4.11.

As formulações do DEM relacionadas a estas leis físicas estão contidas dentro do objeto BLOCK, pois é um procedimento genérico para qualquer geometria de partícula. Mas para os efeitos dos amortecedores SAND usa a rotina DAMP, no qual estão contidos os critérios de servo-controle sobre amortecimento que Cundall [44], [45] formulou para garantir a estabilidade numérica do método, tanto para os parâmetros local e global.

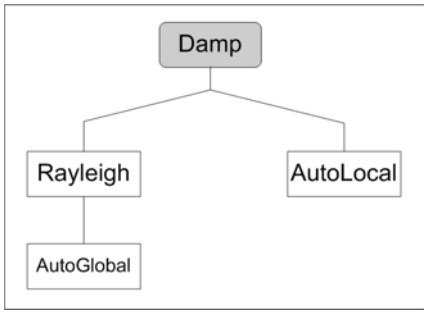


FIGURA 4.13- Hierarquia de classes para a definição dos tipos de amortecimento.

As formulações desta etapa quase não foram modificadas, pois são rotinas de implementação de equações matemáticas comuns sem envolver cálculos iterativos complexos os quais não possam ser resolvidos pela programação tradicional. As formulações desta etapa são simples relativamente e com muito sentido físico que não precisam de muita refinação programática.

4.1.4. Visualização

Seguindo a metodologia do ciclo de cálculo do DEM, só falta a visualização do processo. As principais feições que podem ser mostradas no programa SAND são: os contatos (link), as velocidades, os deslocamento e forças de contato. O ciclo da atualização visual difere da etapa dos cálculos e geralmente usa-se freqüência maior à da etapa numérica. Tipicamente é de 10dt a 500dt para mostrar um ciclo.

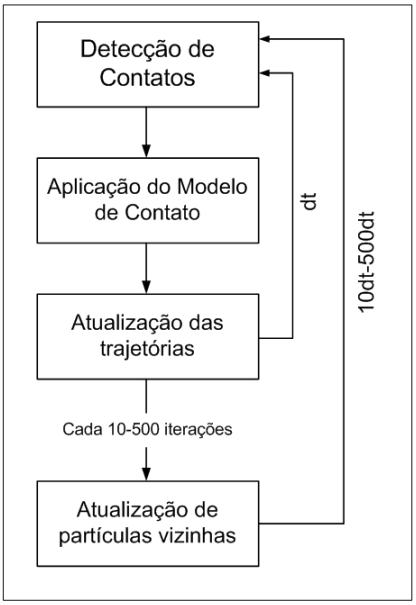


FIGURA 4.14- Esquema da atualização das variáveis no ciclo de cálculo.

Nesta fase fizeram-se mais aplicações do que implantações importantes como, por exemplo, mostrar estratigrafia simbólica para ver efeitos de segregação e dos parâmetros de amortecimento. Esta parte não deixa ser parte importante do processo, pois uma análise de dados com DEM sem parte visual não tem sentido nenhum. O problema radica para aplicações tridimensionais a grande escala o com muito detalhe do fenômeno a ser simulado.

Outras rotinas implementadas no programa SAND estão relacionadas à extração de informação durante a execução do programa como perfis de profundidade de fluxo, velocidades e seguimento de uma partícula de interes. Toda modificação feita no programa tem sido relatada de forma simples neste apartado com a intenção de que sejam constatadas e consideradas em futuros trabalhos com este método.