MAC 0110 – Introdução à Computação para Matemática Aplicada e Computacional

IME – Primeiro Semestre de 2004

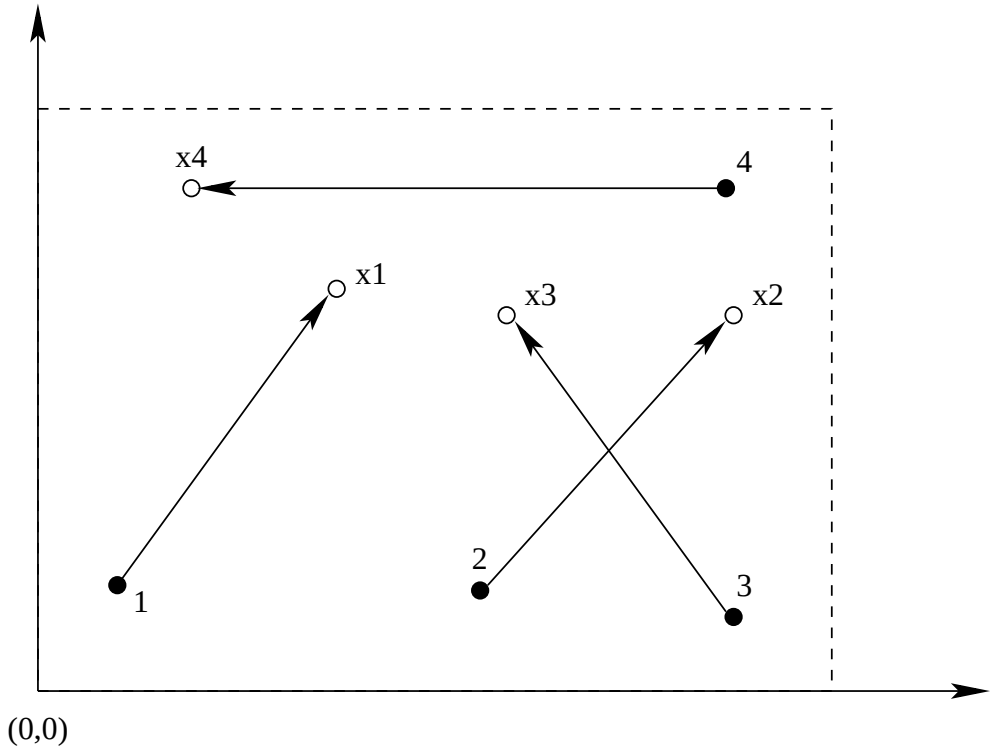
http://panda.ime.usp.br

Segundo Exercício Programa — Entrega: **1 de junho de 2004**

Este EP é uma cópia do que foi proposto como EP3 para os alunos da Poli. Como ele foi muito bem elaborado eu achei conveniente copiá-lo para o nosso curso.

Braço Mecânico

Deseja-se movimentar N objetos dispostos sobre uma mesa usando um braço mecânico.



Este braço deve levar o objeto k para a posição x_k, nesta seqüência de movimentação

- Objeto 1 para a posição x_1;
- Objeto 2 para a posição x_2;
- $\vdots$
- Objeto N para a posição x_N.

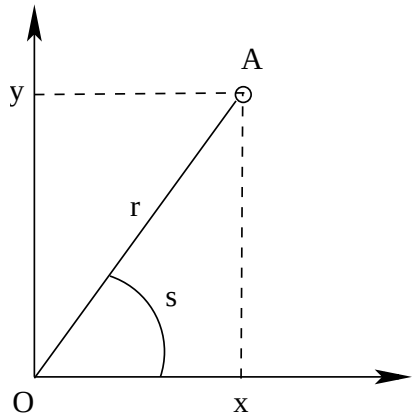
Nota importante: depois de levar o objeto k para a posição x_k, o braço mecânico deve se movimentar até o objeto $k + 1$.

1 Objetivo

O objetivo deste exercício é calcular, para cada k, a distância d_k e o tempo t_k para se levar o objeto k (da sua posição) até a posição x_k e a velocidade média desenvolvida pelo braço mecânico para movimentar os N objetos.

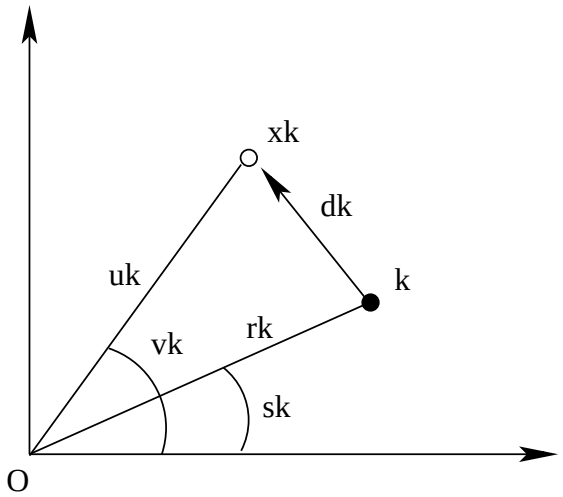
2 Cálculo

Um ponto A no plano pode ser representado por coordenadas cartesianas (x, y). Uma outra forma de representar este ponto A no plano é utilizar coordenadas polares: (r, s), onde r é a distância da origem O até o ponto A e s é o ângulo entre a abscissa e a reta OA.



Vamos considerar que as posições devem ser expressas em *coordenadas polares*.

O braço mecânico inicialmente está na origem, isto é, na posição $(0, 0)$. Depois das N movimentações ele deve ser recolhido novamente para a posição $(0, 0)$.



Considerando a posição do objeto k como sendo (r_k, s_k) e a posição x_k em que o objeto dever ser levado como (u_k, v_k), então a distância entre o objeto k e a posição x_k pode ser calculada usando a lei dos cossenos:

$$d_k = \sqrt{r_k^2 + u_k^2 - 2r_k u_k \cos(v_k - s_k)}.$$

Para calcular o tempo do braço mecânico levar o objeto k para a posição x_k, vamos considerar que a aceleração depende da massa m_k do objeto. Assim, temos:

$$a_k = \frac{AM}{\pi m_k} \text{sen}\left(\frac{\pi m_k}{M}\right)$$

onde M é a massa máxima que o braço suporta e A é a aceleração máxima que o braço pode desenvolver. As constantes M e A dependem das características do braço mecânico. No seu programa, considere π com o valor

$$\pi = 3.1415926536.$$

Observe que, quando a massa do objeto for nula (que é o caso de quando o braço se movimenta para pegar o próximo objeto), o braço mecânico desenvolve aceleração máxima.

Dessa forma, o tempo para o braço levar o objeto k até a posição x_k é

$$t_k = 2\ \sqrt{\frac{d_k}{a_k}}$$

e a velocidade média para movimentar os N objetos é

$$v = \frac{D}{T}$$

onde D é a distância total percorrida pelo braço e T é o tempo total utilizado para movimentar os N objetos.

Note que D e T incluem a distância e o tempo do braço ir da posição x_k até a posição do objeto $k + 1$.

3 Cálculo de erro relativo

Dado um número x e uma aproximação y para x, o *erro* (também chamado de *erro absoluto*) da aproximação y em relação a x é definido como sendo $|y - x|$. Quando a grandeza de x não é próxima da de 1, o erro absoluto pode não ser a maneira mais adequada de medir a qualidade da aproximação y. Por exemplo, os erros absolutos de 1.01 em relação a 1 e de 0.02 em relação a 0.01 são idênticos, mas é claro que a primeira aproximação é muito melhor que a segunda.

Face à limitada avaliação de uma aproximação conferida pelo erro absoluto, tenta-se definir o *erro relativo* de y em relação a x como sendo

$$|y - x|/|x|.$$

Assim, nos dois exemplos anteriores, os erros relativos são respectivamente de 0.01 (ou 1%) e 1 (ou 100%). Contudo, esta definição ainda é incompleta quando $x = 0$. Nestes casos, a divisão por 0 não pode ser realizada e adotam-se valores arbitrários para o erro relativo. No caso de

também ocorrer que $y = 0$, a aproximação certamente é perfeita e adota-se que o erro é 0. No caso de $y \neq 0$, a aproximação é certamente insatisfatória e adota-se o valor arbitrário 1 para o erro relativo. (Na prática, os erros relativos procurados são sempre menores que 1.)

Assim, definimos

$$\text{errorel}(y, x) = \begin{cases} |\frac{y-x}{x}|, & \text{se } x \neq 0 \\ 0, & \text{se } x = 0 = y \\ 1, & \text{se } x = 0 \neq y. \end{cases}$$

4 Cálculo de *sen*(*x*) e *cos*(*x*)

Para calcular uma aproximação de $\text{sen}(x)$ e $\cos(x)$, você deve utilizar obrigatoriamente as seguintes séries:

$$\text{sen}(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + (-1)^k \frac{x^{2k+1}}{(2k+1)!} + \cdots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + (-1)^k \frac{x^{2k}}{(2k)!} + \cdots$$

Para $k = 0, 1, \ldots$, sejam S_k e C_k a soma dos $k + 1$ primeiros termos da série do $\text{sen}(x)$ e do $\cos(x)$, respectivamente. Esta somas devem ser feitas até os menores n e m tais que

$$\text{errorel}(S_{n-1}, S_n) < eps \quad \text{e} \quad \text{errorel}(C_{m-1}, C_m) < eps$$

onde *eps* é um número positivo dado que representa a precisão do cálculo. Use como *eps* o valor 1E-3 (10^{-3}). As aproximações para $\text{sen}(x)$ e $\cos(x)$ que devem ser consideradas pelo seu programa são S_n e C_m, respectivamente.

Note que para calcular $\text{sen}(x)$ e $\cos(x)$ o ângulo x deve estar em radianos.

Observação: seu programa deve **obrigatoriamente** ter uma função de nome seno (respectivamente, cosseno) que recebe um real x e devolve o $\text{sen}(x)$ (respectivamente, $\cos(x)$) usando a série acima. Ele também deve **obrigatoriamente** ter uma função de nome errorel que recebe uma aproximação y, um valor x e devolve o erro relativo $\text{errorel}(y, x)$. Para o cálculo de $\text{errorel}(y, x)$, $\text{sen}(x)$ e $\cos(x)$, seu programa deve **obrigatoriamente** usar as funções errorel, seno e cosseno, respectivamente.

5 Cálculo da raiz quadrada

Para calcular $\sqrt{x}$ quando $x > 0$, você deve utilizar a seguinte recorrência:

$$\begin{cases} b_0 = x \\ b_i = \frac{1}{2}\left(b_{i-1} + \frac{x}{b_{i-1}}\right) & i = 1, 2, \cdots \end{cases}$$

Por exemplo, os termos b_1 e b_2 são calculados da seguinte forma:

$$b_1 = \frac{1}{2}\left(b_0 + \frac{x}{b_0}\right) = \frac{1}{2}\left(x + \frac{x}{x}\right) = \frac{x+1}{2}$$

$$b_2 = \frac{1}{2}\left(b_1 + \frac{x}{b_1}\right) = \frac{1}{2}\left(\frac{x+1}{2} + \frac{2x}{x+1}\right).$$

A partir de b_2, obtemos b_3 e assim por diante.

Este processo deve ser repetido até o primeiro n tal que errorel$(b_n, b_{n+1}) < eps$, onde eps é um número positivo dado que representa a precisão do cálculo da raiz. A aproximação obtida para $\sqrt{x}$ será b_{n+1}. Use como eps o valor 1E-3 (10^{-3}).

Cuidado!!! Não aplique a recorrência para $x = 0$, ou ocorrerá uma divisão por zero!!!

Observação: seu programa deve **obrigatoriamente** ter uma função de nome raiz que recebe um real x e devolve a sua raiz quadrada usando a recorrência acima. Ele também deve **obrigatoriamente** ter uma função de nome errorel que recebe uma aproximação y, um valor x e devolve o erro relativo errorel(y, x). Para o cálculo de errorel(y, x), raiz(x), seu programa deve **obrigatoriamente** usar as funções errorel e raiz, respectivamente.

6 O que seu programa deve fazer

Seu programa deve ler:

- M, a maior massa que o braço pode suportar;
- A, a aceleração máxima do braço mecânico;
- N, o número de objetos a serem transportados;
- r_k, s_k, u_k, v_k, m_k, para $k = 1, 2, \cdots, N$, onde (r_k, s_k) representa a posição do objeto k (note que s_k é um ângulo em **graus**) e (u_k, v_k) representa a posição x_k (note que v_k é um ângulo em **graus**) e m_k a massa do objeto;

O seu programa **NÃO** precisa testar a consistência dos dados de entrada, isto é, você pode supor que o usuário irá digitar os dados de uma forma consistente:

- a massa máxima e a aceleração máxima são maiores do que zero;
- qualquer massa de entrada é um número positivo e **menor** que a massa máxima;
- qualquer ângulo está no intervalo de $[0, 360[$;
- qualquer distância é maior ou igual a zero.

Seu programa deve imprimir:

- cada distância d_k e cada tempo t_k; e

- a velocidade média D/T que o braço desenvolveu para transportar os N objetos. Adotar velocidade média zero se $T = 0$. Lembre-se de que seu programa deve considerar as distâncias e os tempos gastos entre

  1. a origem e o primeiro objeto;
  2. o objeto k e a posição x_k;
  3. a posição x_k e o objeto $k + 1$; e
  4. o último objeto até a origem.

Note que seu programa deverá ler os ângulos em graus e transformá-los em radianos para serem usados na fórmula do sen(x) e cos(x).

7 Dados para teste

Teste seu programa para pelo menos os seguintes dados de entrada:

- $M = 20\ kg$
  $A = 0.01\ m/s^2$

  $N = 1$

  $r_1 = 0.08\ m$, $s_1 = 0^o$
  $u_1 = 0.08\ m$, $v_1 = 0^o$
  $m_1 = 4\ kg$

- $M = 20\ kg$
  $A = 0.07\ m/s^2$

  $N = 4$

$r_1 = 0\ m$, $s_1 = 0^o$	$r_2 = 0.20\ m$, $s_1 = 90^o$	$r_3 = 0.13\ m$, $s_3 = 120^o$	$r_4 = 0.01\ m$, $s_4 = 20^o$
$u_1 = 0.07\ m$, $v_1 = 0^o$	$u_2 = 0.07\ m$, $v_2 = 30^o$	$u_3 = 0.08\ m$, $v_3 = 200^o$	$u_4 = 0.12\ m$, $v_4 = 180^o$
$m_1 = 14\ kg$	$m_2 = 19\ kg$	$m_3 = 9\ kg$	$m_4 = 17\ kg$

- $M = 20\ kg$
  $A = 0.01\ m/s^2$

  $N = 1$

  $r_1 = 0\ m$, $s_1 = 0^o$
  $u_1 = 0\ m$, $v_1 = 0^o$
  $m_1 = 4\ kg$

- $M = 10\ kg$
  $A = 0.04\ m/s^2$

  $N = 2$

  $r_1 = 0.07\ m,\ s_1 = 60^o$ $\quad r_2 = 0.07\ m,\ s_2 = 240^o$
  $u_1 = 0.07\ m,\ v_1 = 0^o$ $\quad u_2 = 0.07\ m,\ v_2 = 300^o$
  $m_1 = 4\ kg$ $\quad m_2 = 4\ kg$

- $M = 10\ kg$
  $A = 0.04\ m/s^2$

  $N = 2$

  $r_1 = 0.7\ m,\ s_1 = 60^o$ $\quad r_2 = 0.5\ m,\ s_2 = 90^o$
  $u_1 = 0.7\ m,\ v_1 = 30^o$ $\quad u_2 = 0.5\ m,\ v_2 = 0^o$
  $m_1 = 0.04\ kg$ $\quad m_2 = 1\ kg$

8 Observações importantes

1. O uso das funções da biblioteca `<math.h>` não é permitido neste exercício programa. Agora, você pode utilizá-las (em uma fase de testes) para comparar os resultados das suas funções matemáticas. A versão final do seu programa (que você entregará no panda) **NÃO** deverá conter funções desta biblioteca.

2. A definição e o uso de outras funções convenientes são permitidos neste exercício.

3. Faça seu programa com extensão "`.c`". Não serão aceitos programas com outros tipos de extensão tais como "`.cpp`", "`.exe`".

4. Uma novidade na correção deste exercício programa é o desconto de pontos para programas que emitem "warnings" na compilação. Um "warning" é uma aviso do compilador de que alguma coisa pode não estar correto no seu programa. Estes avisos podem ser uma boa indicação de erros de lógica. Por exemplo,

   ```
   if (a=b) {
     [. . .]
   }
   ```

   resulta em um "warning", e provavelmente é um erro (atribuição ao invés de comparação). Dessa forma, **procure** eliminar todas as fontes de "warnings" de seu programa.

   Para ativar a detecção de "warnings" no LCCWin escolha a sub opção "Configuration" da opção "Project" do menu, clique na aba "Compiler" e escolha a opção "All" para o Warning level.

5. Se você acha muito chato ficar digitando um monte de dados de entrada, é possível fazer com que seu programa leia dados de um arquivo. Embora não seja uma exigência deste exercício, fica a seu critério implementar leitura de arquivo no seu programa. Existe uma receita de como fazer leitura de arquivos em `http://www.ime.usp.br/~mac2166/ep3/leitura-arquivo.htm`

9 Executável

Um executável para linux e outro para windows deste exercício programa podem ser encontrados em `http://www.ime.usp.br/~mac2166/ep3/executaveis/`. Caso você tenha dúvidas sobre qual deve ser o comportamento do seu programa em alguma situação, veja como se comportam os executáveis.

10 Nota explicativa para o cálculo do tempo

Considerando que na primeira metade do percurso (o braço percorre uma distância $\frac{d_k}{2}$) a aceleração é positiva, temos que, pelas equações de Torricelli e da velocidade

$$v_k^2 = a_k d_k \quad \text{e} \quad v_k = a_k t'_k.$$

Logo, temos

$$t'_k = \sqrt{\frac{d_k}{a_k}}.$$

Supondo que a aceleração na segunda metade do percurso seja negativa (de mesma intensidade), obtemos que o tempo de movimentar o braço de uma distância d_k é

$$t_k = 2t'_k = 2\ \sqrt{\frac{d_k}{a_k}}.$$