Exercices programmation – Matlab 4 +corr

Intégration

Calculez par la méthode des trapèzes (cumtrapz) l'intégrale ***yc*** d'un sinus dans l'intervalle **x**= 0 à 2*pi. La taille du pas est un paramètre `stepSz` initialement à pi/3.

```
stepSz=pi/3;
x = …
f = @(x) …
y = f(x)
yc= cumtrapz(…)
```

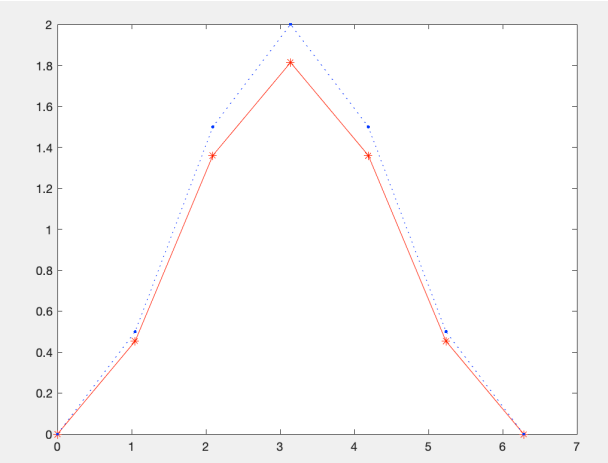
Calculez symboliquement **sInt**, l'intégrale (int) d'un sinus. Puis l'évaluer dans l'intervalle **xx**= 0 à 2*pi. Pour cela convertir le résultat de l'intégration symbolique en une fonction **af** anonyme matlab avec (matlabFunction) et évaluer cette fonction pour l'intervalle **xx**.

```
syms xx
ff = sin(xx)

sInt = …
af = matlabFunction(…)
y2 = …
```

Afficher les deux courbes et comparer. Faites varier la taille du pas pour voir son influence.

```
Plot(…)
```

```
stepSz=pi/3;
x= 0:stepSz:2*pi ;
f=@(x) sin(x);

y1= f(x);
yc=cumtrapz(x,y1);

syms xx
ff= sin(xx);
SymInt=int(ff)

af=matlabFunction(SymInt)
y2=af(x)-af(x(1));

plot(x,yc,'-r*',x,y2,':b.')
```



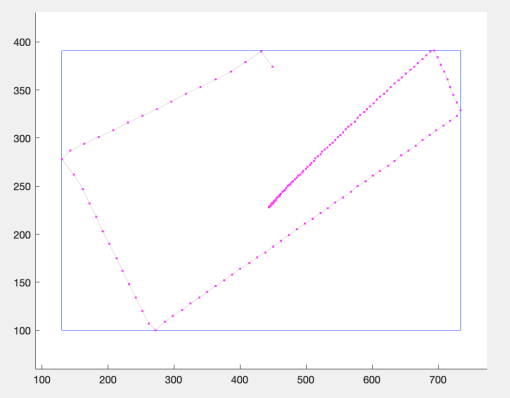
Dérivation - Billard

```
X=[449 432 408 386 363 340 318 296 274 252 230 208 186 164 143 130 148 162 172
182 192 202 213 222 232 242 252 262 272 286 298 312 325 338 350 363 376 388 400
414 426 438 450 462 474 486 498 510 522 533 544 556 568 578 590 601 612 624 634
644 655 666 676 687 697 708 718 728 734 728 723 718 714 709 704 699 694 688 682
676 669 664 658 651 645 640 634 628 623 618 612 607 602 597 592 588 582 578 574
568 564 560 556 552 548 544 540 536 532 528 524 522 518 514 512 508 506 502 500
496 494 491 488 486 483 480 478 476 474 472 470 468 466 464 462 460 460 458 456
455 454 452 452 450 450 448 448 446 446 446 444 444 444 444 444];

Y=[374 390 379 369 361 353 346 338 330 323 316 308 301 294 287 278 262 247 232
218 203 190 175 162 148 134 120 107 100 109 115 121 128 134 140 146 152 158 164
170 176 181 187 193 199 205 211 216 222 227 233 238 244 250 255 261 266 271 276
282 287 292 298 303 308 313 318 323 329 337 345 353 361 369 376 384 391 390 386
382 378 374 371 367 363 360 357 353 349 346 343 340 336 333 330 327 324 321 317
314 312 309 306 303 301 298 295 293 290 288 286 283 281 279 276 274 272 270 268
266 264 262 261 259 257 255 254 252 251 250 248 246 245 244 242 241 240 239 238
237 235 235 234 233 232 232 231 230 230 229 229 228 228 228 228];
```

Les vecteurs X et Y représentent les coordonnées XY (en pixels) d'une boule de billard en mouvement. Cette boule touche les 4 bords du billard.

Déterminer la valeur de ces quatre boards

```
Xmin= …
Xmax= …
Ymin= …
Ymax= …
```



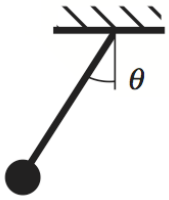
Afficher ces bords
Afficher la trace de la boule

Calculer la distance parcourue par la balle (en pixels)
`Dist=…`

```
Xmax=max(X); Xmin=min(X);
Ymax=max(Y); Ymin=min(Y);

Xframe=[Xmin Xmax Xmax Xmin Xmin];
Yframe=[Ymin Ymin Ymax Ymax Ymin];
plot(Xframe,Yframe,'b-');

% Compute distances between pts
D=sqrt((diff(X).*diff(X))+(diff(Y).*diff(Y)));

% Compute the ball distance, i.e. the sum of all segments
Dist=sum(D)

plot(X,Y,'MarkerFaceColor',[1 0 1],'…
      MarkerEdgeColor',[1 0 1],'MarkerSize',5,'Marker','.',…
      'Color',[0.9 0.8 0.8]);
```

Dist = 1470.7 px

Simulation d'un pendule



Exemple tiré du cours 2e, simulation d'un pendule sans frottement.
La longueur du bras L est de 1m.

Le modèle à intégrer à 2 états : theta et dtheta -> x(1) et x(2)
avec

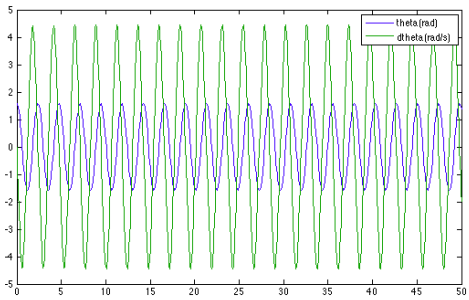
```
g = 9.81; L = 1;

% Définir le modèle à intégrer comme une @ function
dxdt = @(t,x) [x(2); -g*sin(x(1))/L];
```

Conditions initiales 1
Le pendule est à l'horizontale (pi/2), vitesse = 0

```
% cond
x0 = [pi/2 0];
```

Temps de simulation de 0 à 50 secondes.

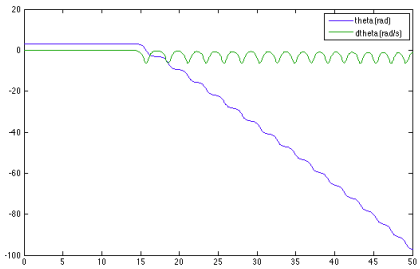
```
ts = 0;  tf = 50;

[t x]= ode45(dxdt,[ts tf],x0);% integration

plot(t,x);        % affichage
legend('theta (rad)','dtheta (rad/s)')
```



Conditions initiales 2
Le pendule est à la verticale (pi), vitesse = 0

```
x0 = [pi 0];
```

Refaire la simulation avec les nouvelles conditions initiales. Le pendule devrait rester en position verticale (instable), mais l'accumulation des erreurs numériques suffi à perturber la simulation.



La simulation peut-être améliorée en spécifiant les tolérances aux erreurs relatives et absolues via odeset()

```
odeopt = odeset('RelTol',1e-10,'AbsTol',1e-10);
```

Refaire la simulation avec ces options

```
[t x] = ode45(dxdt,[ts tf],x0,odeopt);
```