UNIVERSITÀ DEGLI STUDI DI BRESCIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN

INGEGNERIA INFORMATICA

Realizzazione di animazioni con HTML5 Canvas per la comprensione dei concetti di derivata prima e funzione integrale.

Relatrice

Prof.ssa Paola Gervasio

Laureando

Claudio Catterina

Matr. 85957

Anno Accademico 2014/2015

Indice

INTRODUZIONE

Obiettivi della tesi

L’obiettivo di questa tesi è quello di sviluppare HTML5 Canvas che aiutino gli studenti del corso di Analisi Matematica 1 nell’apprendimento dei concetti di derivata prima e integrale.

Per quanto concerne la derivata prima di una funzione si vuole mostrare dal punto di vista geometrico il significato di derivata prima in un punto come limite del rapporto incrementale, la costruzione della funzione derivata, la corrispondenza tra crescenza/decrescenza di una funzione ed il segno della sua funzione derivata prima.

Per quanto riguarda il concetto di integrale, si vuole dare l’interpretazione geometrica alla definizione di funzione integrale di una funzione limitata assegnata. Inoltre si vuole mostrare graficamente che in genere l’integrale improprio su R di una funzione dispari non è nullo.

I canvas sviluppati in linguaggio HTML5 saranno di supporto alla spiegazione del docente in aula e potranno essere utilizzati dallo studente per verificare in autonomia se egli ha compreso i concetti spiegati. Lo studente potrà modificare la funzione su cui operare e i parametri per il disegno, ovvero gli intervalli di valutazione e rappresentazione della funzione e il punto in cui calcolare la derivata prima.

Non è obiettivo dei programmi scritti calcolare derivate o integrali. Sappiamo infatti che ogni passaggio al limite non è realizzabile con calcolatori digitali. Tuttavia, basandosi su semplici formule numeriche (come il rapporto incrementale in avanti o all’indietro del primo ordine e la formula di quadratura dei trapezi) i programmi forniscono un output grafico dinamico che aiuta lo studente a capire meglio i concetti di derivata e integrale.

Capitolo 1:

CONCETTI DI ANALISI 1

Nel presente capitolo vengono presntati i concetti dell'analisi matematica 1 su cui si basano le animazioni sviluppate nel lavoro di tesi. Per una spiegazione più ampia di tali concetti rimandiamo a **[1]**.

1.1 Derivata prima in un punto

Sia $f\colon \mathrm{dom}(f) \subseteq \mathbb{R} \to \mathbb{R}$; sia $x_0 \in \mathrm{dom}(f)$ e punto di accumulazione per $\mathrm{dom}(f)$.

Def. Se esiste (finito o infinito) il limite $\lim_{x \to x_0} \frac{f(x)-f(x_0)}{x-x_0}$, poniamo

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x)-f(x_0)}{x-x_0}$$

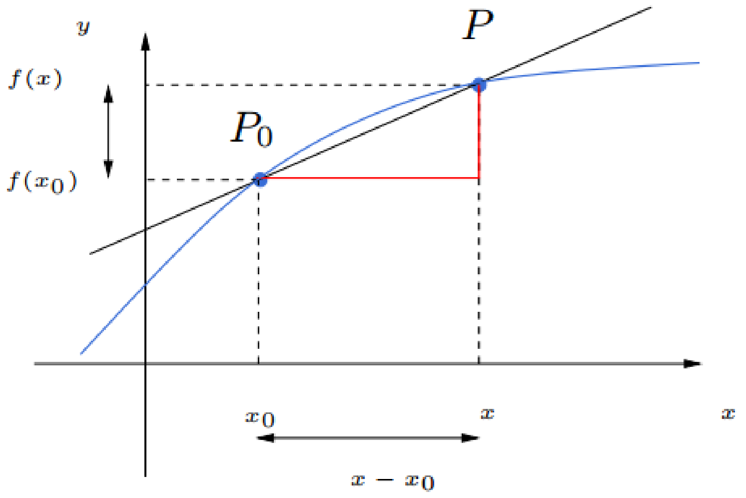
e chiamiamo derivata prima di f in x_0 il numero $f'(x_0) \in \overline{\mathbb{R}}$.

Se $f'(x_0) \in \mathbb{R}$ (cioè è finito), diciamo che f è derivabile in x_0.

$\frac{f(x)-f(x_0)}{x-x_0}$ è detto rapporto incrementale, quindi si dice anche che la derivata prima nel punto x_0 è il limite del rapporto incrementale di f quando x tende a x_0.

Interpretazione geometrica

Sia $x - x_0$ l'incremento (> 0 o < 0) della variabile indipendente x e $f(x) - f(x_0)$ l'incremento (> 0 o < 0) della variabile dipendente y. Sia f derivabile in x_0.



$$P_0 = (x_0, f(x_0)),$$
$$P = (x, f(x))$$

Il coefficiente angolare della retta passante per i punti P_0 e P è

$m_{(x_0,x)} = \frac{f(x)-f(x_0)}{x-x_0}$ (=rapporto incrementale)

Il rapporto incrementale di f in x_0 dipende sia da x che da x_0.

Geometricamente, la derivata prima di f nel punto x_0 (ovvero $f'(x_0)$) è il limite per $x \to x_0$ dei coefficienti angolari $m_{(x_0,x)}$ delle rette passanti per i punti $P_0 = (x_0, f(x_0))$ e $P = (x, f(x))$.

La derivata prima di f in x_0 è il coefficiente angolare della retta tangente alla curva $y = f(x)$ nel punto x_0.

Geometricamente , una funzione f è derivabile in x_0 se esiste la retta tangente ad f nel punto x_0 e questa non è una retta verticale.

1.1.2 La funzione derivata prima f ′(x)

Def. Sia I un intervallo contenuto nel dominio di f. Se f è derivabile in ogni punto di I, diciamo che f è derivabile in I .

Def. Poniamo $\mathrm{dom}(f') = \{x \in \mathrm{dom}(f) \colon f$ è derivabile in $x\}$ .Definiamo funzione derivata prima di f la funzione che associa ad ogni $x \in \mathrm{dom}(f')$ il valore $f'(x)$:

$$f' \colon \quad \mathrm{dom}(f') \subseteq \mathbb{R} \to \mathbb{R}$$
$$x \mapsto f'(x)$$

1.1.3 Criterio del segno della derivata prima

Teorema.

Sia $I \subseteq \mathrm{dom}(f)$ un intervallo e sia f derivabile su I. Allora

$f'(x) \geq 0,\ \forall x \in I \qquad \Leftrightarrow \qquad f$ è crescente su I e

$f'(x) > 0,\ \forall x \in I \qquad \Rightarrow \qquad f$ è strettamente crescente su I.

1.2 Funzione integrale

Sia f definita su un intervallo $I \subset \mathbb{R}$, limitata e localmente integrabile secondo Riemann.

Sia $x_0 \in I$ fissato. Definiamo Funzione integrale di f la funzione

$$F_{x_0}(x) = \int_{x_0}^{x} f(t)dt$$

Oss. Per la definizione di integrale definito, si ha

$$F_{x_0}(x_0) = \int_{x_0}^{x_0} f(t)dt = 0.$$

1.3 Integrale su $\mathbb{R} = (-\infty, \infty)$

Consideriamo $f \colon \mathbb{R} \to \mathbb{R}$, limitata e localmente integrabile secondo Riemann e vogliamo calcolare

$$\int_{-\infty}^{+\infty} f(x)dx. \tag{1}$$

Sia $c \in \mathbb{R}$, definiamo

$$\int_{-\infty}^{+\infty} f(x)dx \quad = \lim_{\substack{a\to-\infty \\ b\to-\infty}} \int_a^b f(x)dx = \tag{2}$$

$$= \lim_{a\to-\infty} \int_a^c f(x)dx + \lim_{b\to+\infty} \int_c^b f(x)dx$$

e diciamo che l'integrale improprio (1) è convergente (o esiste finito) SE esistono finiti i due integrali impropri che compaiono in (2).

Le due variabili a e b per i limiti sono a priori diverse, i due limiti devono essere indipendenti l'uno dall'altro.

Consideriamo ad esempio la funzione $f(x) = \frac{2x}{1+x^2}$.

$$\ell = \int_{-\infty}^{+\infty} \frac{2x}{1+x^2} dx \quad = \lim_{\substack{a\to-\infty \\ b\to-\infty}} \int_a^b \frac{2x}{1+x^2} dx = \lim_{\substack{a\to-\infty \\ b\to-\infty}} [\log(1+x^2)]_a^b$$

$$= \lim_{\substack{a\to-\infty \\ b\to-\infty}} \log \frac{1+b^2}{1+a^2}$$

Ora, se $a = -b$ si ha

$$\ell = \lim_{b\to+\infty} \log \frac{1+b^2}{1+b^2} = 0$$

ma se se $a = -b^2$ si ha

$$\ell = \lim_{b\to+\infty} \log \frac{1+b^2}{1+b^4} = -\infty$$

Il valore del limite dipende da quanto veloci a e b vanno all'infinito. Il Valore Principale di Cauchy di un integrale improprio è l'integrale improprio calcolato con $a = -b$, e si scrive come

$$V.P. \quad \int_{-\infty}^{\infty} f(x)dx = \lim_{b\to+\infty} \int_{-b}^{b} f(x)dx. \tag{3}$$

Quindi

$$\int_{-\infty}^{\infty} \frac{2x}{1+x^2} dx \quad \grave{e}\ una\ forma\ indeterminata,$$

dipende dalla velocità con cui a e b vanno all'infinito, mentre

$$V.P. \int_{-\infty}^{\infty} \frac{2x}{1+x^2} dx = 0.$$

Capitolo 2:

CANVAS HTML5

2.1 I Canvas

Il canvas è un elemento compreso in HTML5 **[3]** e come ogni altro elemento HTML viene rappresentato da un tag, in questo caso il tag “<canvas>”.

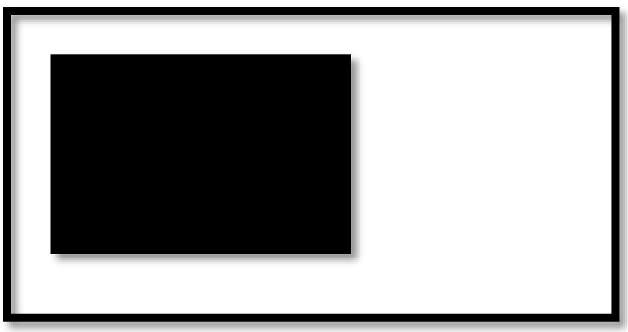
Questo elemento permette il rendering dinamico di una bitmap, ossia definisce una porzione di spazio di una pagina web nella quale, attraverso l’utilizzo di Javascript e specifiche API è possibile tracciare forme geometriche come cerchi, linee o rettangoli, inserire immagini o scrivere del testo.

Le possibili applicazioni sono molteplici, partendo da queste forme di base è possibile sviluppare grafici complessi, composizioni di immagini, videogiochi, o come nel caso di questo progetto, animazioni.

Esempio:

```
<canvas width="300px" height="300px" id="demo_canvas">
      Contenuto da mostrare in caso il canvas non sia supportato.
</canvas>
```

```
var canvas = document.getElementById("demo_canvas");
var context = canvas.getContext("2d");
context.beginPath();
context.fillStyle=”Black”;
context.fillRect(20,20,150,100);
context.fill()
```



2.2 I motivi di questa scelta

La scelta di utilizzare i canvas per questo progetto non è stata affatto ovvia, esistono infatti diverse alternative per disegnare e creare animazioni in pagine web, le principali sono SVG, Flash e Applet:

- SVG (Scalable Vector Graphics) è un modello di grafica in modalità mantenuta permanente all'interno di un modello in memoria che può essere modificato mediante risultati di codice durante il re-rendering.
- Flash è un software che permette di realizzare grafici e animazioni composti principalmente da grafici vettoriali.
- Le Applet Java sono programmi scritti in linguaggio Java che possono essere eseguiti da un Web Browser.

Le caratteristiche principali di Canvas e in parte di SVG che hanno portato all'immediata esclusione di Flash e Applet sono l'alto livello di compatibilità e la semplicità d'uso, infatti Canvas non solo è compatibile con tutti i principali web browser per pc ma anche con browser mobile, il tutto senza l'utilizzo di plug-in esterni.

Per quanto riguarda invece il confronto con SVG, esso è stato principalmente sulle prestazioni che, seppur molto simili, hanno favorito ancora una volta Canvas. (Per maggiori informazioni su questo confronto rimandiamo a **[4]**.

Figura 1 - Compatibilità Canvas HTML5

Capitolo 3:
ANALISI E PROGETTAZIONE

3.1 Analisi dei requisiti

In questa fase, preliminare allo sviluppo del software, sono state definite tutte le funzionalità che l'applicazione deve offrire ed i vincoli imposti portando quindi alla stesura di una specifica dei requisiti del software:

- Scopo
  - Sviluppare un'applicazione che generi animazioni utili all'apprendimento dei concetti di derivata prima e funzione integrale.

- Vincoli
  - L'applicazione dev'essere eseguibile da browser.
  - Non deve presentare grossi limiti di compatibilità in quanto usufruibile da un'ampia gamma di utenti aventi sistemi differenti.

- Requisiti
  - Per ogni tipo di animazione il sistema deve consentire la scelta dinamica di funzioni matematiche ad una variabile (interamente definite o definite a tratti).
  - Per ogni tipo di animazione il sistema deve permettere la scelta di un range di valori per l'asse delle ascisse e di uno (o eventualmente due differenti per due grafici), per l'asse delle ordinate.

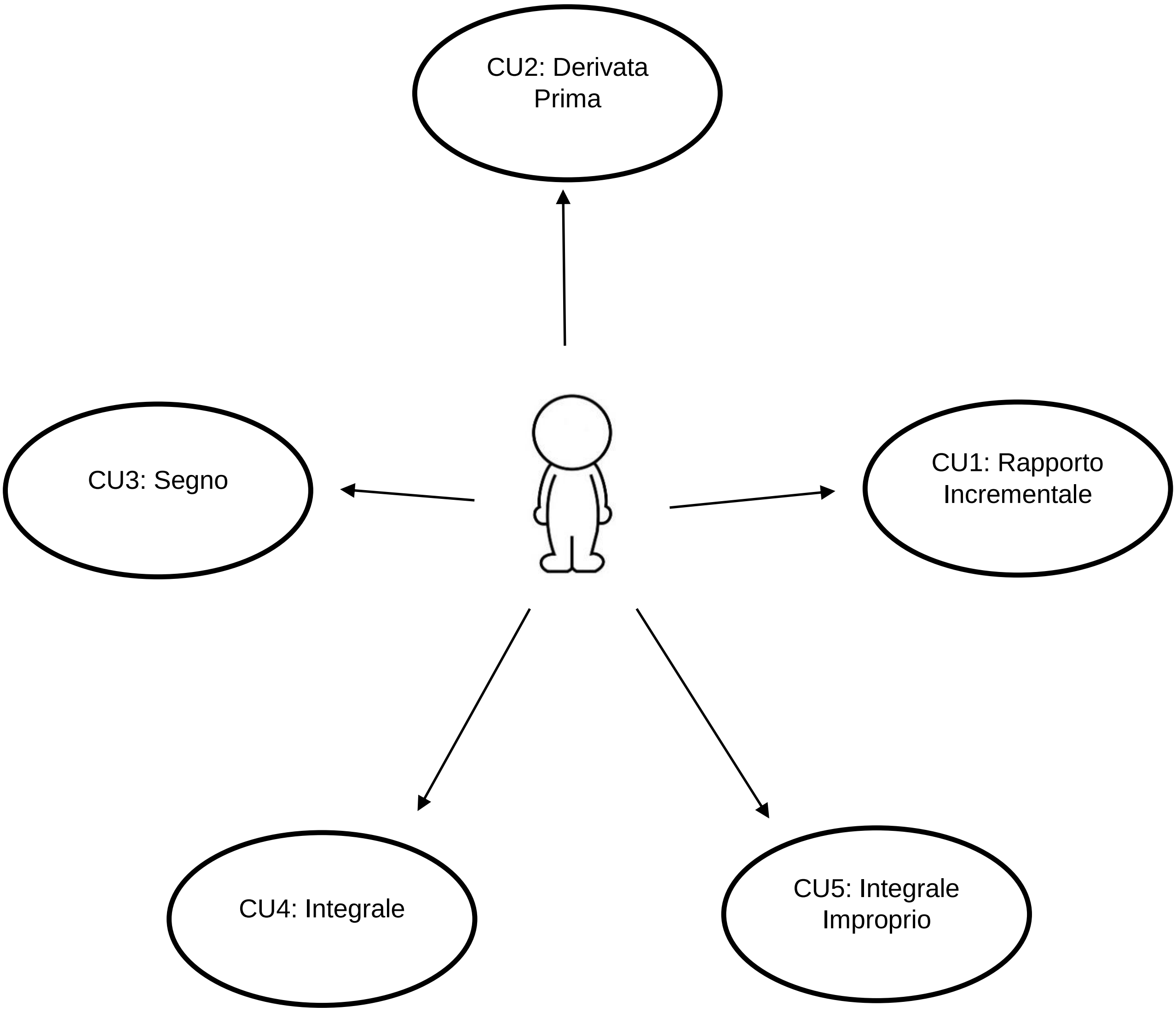
- Il sistema deve presentare, per ogni animazione una descrizione e delle istruzioni di utilizzo.
- Il sistema deve generare un'animazione che presi due punti, x0 e x, disegni la retta passante per i due punti al variare di x fino a far tendere x a x0.
- Il sistema deve generare un'animazione che si sposti lungo i punti della funzione inserita e disegni per ogni punto la tangente in quel punto alla funzione ed il valore della derivata prima in un secondo grafico.
- Il sistema deve generare un'animazione che si sposti lungo i punti della funzione inserita e disegni per ogni punto la tangente in quel punto alla funzione ed il valore della derivata prima mostrando graficamente la corrispondenza tra crescenza/decrescenza della funzione ed il segno della sua funzione derivata prima.
- Il sistema deve generare un'animazione che mostri il concetto di integrale spostandosi lungo i punti della funzione, colorando l'area sottesa ad essa e disegnando il valore della funzione integrale in un secondo grafico.
- Il sistema deve generare un'animazione che mostri il concetto di integrale improprio spostandosi contemporaneamente da 0 ad infinito e da 0 a –infinito colorando le aree sottese alla funzione e disegnando la somma di esse in un secondo grafico.
    - Vengono accettati solo range di valori dell'asse delle ascisse simmetrici.
    - E' permesso scegliere fra due velocità di integrazione diverse (a=-b o a=-b^2)

3.2 Casi d'uso

Successivamente alla stesura della specifica dei requisiti sono stati analizzati i casi d'uso, in questo caso i 5 casi d'uso corrispondenti alle 5 animazioni sono quasi del tutto simili e per questo ne viene riportato soltanto uno:

CU2: Derivata prima
Attore: Utente
Precondizione: E' aperta la pagina relativa alla derivata prima e non è in esecuzione alcuna animazione.
Sequenza Eventi: 1. L'attore inserisce i dati nei campi disponibili (funzione, xmin, xmax, ymin, ymax); al termine preme il bottone "disegna". 2. Il sistema controlla i dati inseriti, se sono corretti avvia l'animazione e disabilita il pulsante "disegna" in modo da evitare che l'attore possa avviare un'altra animazione in contemporanea alla prima. 3. Il sistema conclude l'animazione e riabilita il pulsante "disegna". 4. Fine. Si ritorna alla situazione iniziale.
Scenario alternativo: Se al punto 1 l'attore inserisce i dati in modo errato, il sistema non avvia l'animazione e mostra un pannello di allarme specificando quali dati sono errati.

Schema casi d’uso:



3.3 Progettazione e scelte di sviluppo

Nella fase di progettazione, si è scelta la tecnologia da utilizzare per le animazioni, il linguaggio di programmazione e gli strumenti da utilizzare per lo stesura del codice.

La scelta della tecnologia da utilizzare è ricaduta sui Canvas HTML5, principalmente per la vasta compatibilità offerta, come già approfondito nel capitolo 2, e di conseguenza la scelta di javascript come linguaggio di programmazione per modificare dinamicamente i contenuti dei Canvas è stata obbligatoria in quanto unico linguaggio disponibile per tale scopo.

Gli strumenti utilizzati per la stesura del codice e scelti in questa fase sono stati:

“Sublime Text 3” come editor **[5]**;

il package “DocBlockr” **[6]** (incluso in Sublime Text 3) per facilitare la stesura dei commenti e renderla compatibile con i principali generatori di documentazione;

“doxx” **[7]**, come generatore di documentazione.

Capitolo 4:

STESURA DEL CODICE

4.1 Il codice

Il codice sviluppato è suddiviso in tre principali file javascript:

1. *Function.js*
2. *Animation.js*
3. *DrawUtilities.js*

Function.js contiene funzioni che si occupano di:

- Studiare il dominio e disegnare il grafico della funzione statica.
- Definire e disegnare gli assi, calcolando quali unità è più corretto visualizzare.
- Studiare i valori dell'integrale, dell'integrale improprio e della derivata nel range di valori delle x, in modo da definire degli assi appropriati per il canvas in cui viene disegnata la funzione integrale e la funzione derivata prima.

Di seguito sono riportate le funzioni e gli estratti di codice più significativi:

- drawFunc()

```
/**
     * Disegna la funzione statica sul canvas
     * @param  {context}  ctx   context del canvas
     * @param  {axes}     axes  assi su cui disegnare la funzione
     * @param  {String}   func  funzione da disegnare
     * @return {dom}            dominio della funzione
     */
    function drawFunc (ctx,axes,func)
    {
      var dom={};
      dom.last_pixel=WIDTH;
      dom.first_pixel=0;
      var first_found = false;
      for (var i=0;i<=WIDTH;i++) {
        px = i;
        xx = i+axes.xmin_px;
        yy = axes.scale_y*Parser.evaluate(func, { x: (xx/axes.scale_x) });
        //se incontro valori non definiti coloro in grigio la parte di grafico
        if (isNaN(yy)){
          ctx.beginPath();
          ctx.fillStyle =  "rgba(11, 13, 15, 0.3)";
          ctx.fillRect(px,0,1,HEIGHT);
```

```
        ctx.fill();
        ctx.closePath();
      }else{
        if (!first_found)
          dom.first_pixel=px;
        first_found=true;
        dom.last_pixel=px;
      }
      ctx.beginPath();
      ctx.fillStyle =  "black";
      //implementazione a punti
      ctx.arc(px,HEIGHT-axes.ymin_px-yy,2,0,2*Math.PI);
      ctx.fill();
      ctx.closePath();
    }
    return dom;}
```

- Estratto di drawAxes() utile a disegnare l’asse delle ascisse

```
    ctx.beginPath();
    ctx.strokeStyle="black";
    ctx.lineWidth=1;
    var x_px=axes.x0!=-1?axes.x0:HEIGHT-2;
    //disegno l'asse privo di unità
    ctx.moveTo(0,x_px);
    ctx.lineTo(WIDTH,x_px);
    ctx.stroke();
    //scorro ogni pixel del canvas
    for(var px=1;px<=WIDTH;px++){
      var xx=(px+axes.xmin_px)/axes.scale_x;
      var xx_pre=(px-1+axes.xmin_px)/axes.scale_x;
      var xx_post=(px+1+axes.xmin_px)/axes.scale_x;
      //individuo il pixel piÃ¹ vicino allo step da disegnare
      if (Math.abs(xx%axis_int.step)<Math.abs(xx_pre%axis_int.step) && Math.abs(xx%axis_int.step)<Math.abs(xx_post%axis_int.step)){
        ctx.beginPath();
        ctx.fillStyle="black";
        ctx.font="10px Georgia black";
        //disegno una linea di 4 pixel
        ctx.moveTo(px,x_px+2);
        ctx.lineTo(px,x_px-2);
        ctx.stroke();
        ctx.beginPath();
        //scrivo il valore corrispondente
        if (axis_int.decimal>=0)
          ctx.fillText(""+Math.round(xx*Math.pow(10,axis_int.decimal))/Math.pow(10,axis_int.decimal),px-4,x_px-5 );
        else
          ctx.fillText(""+Math.round(xx),px-4,x_px-5 );
          ctx.fill();
        }
      }
```

- studyIntegral() utile a raccogliere informazioni sui valori dell’integrale prima di procedere all’animazione.

```
function studyIntegral(func,axes){
    var integral={};
    integral.total_area=0;
    integral.max_area=0;
```

```
      integral.values=new Array()
      integral.y_axes_correction=1;
      for (var i=axes.xmin_px;i<axes.xmax*axes.scale_x;i++){
        if (!isNaN(Parser.evaluate(func, { x: i/axes.scale_x })) && isFinite(Parser.evaluate(func, { x: i/axes.scale_x })))
          integral.total_area+=1/axes.scale_x*Parser.evaluate(func, { x: i/axes.scale_x });
        integral.max_area=Math.abs(integral.total_area)>Math.abs(integral.max_area)?Math.abs(integral.total_area):integral.max_area;
        integral.values[i-axes.xmin_px]=integral.total_area;
      }

      if((HEIGHT/2)/axes.scale_y<integral.max_area){
        integral.y_axes_correction=Math.ceil(integral.max_area/((HEIGHT/2)/axes.scale_y))
      }
      return integral;
    }
```

Animation.js è invece il cuore dell'applicazione in quanto contiene le funzioni utili a generare tutte le animazioni, di seguito sono riportati i principali spezzoni di codice:

- drawDerivate()

```
          ctx.beginPath();
          ctx.clearRect(0,0,WIDTH,HEIGHT);
          ctx.stroke();
          ctx.lineWidth=2;

          //derivata nel punto i (f(i+0.00001)-f(i))/0.00001
          var derivative=(Parser.evaluate(func, { x: xx/axes.scale_x+0.0000001 }) - Parser.evaluate(func, { x: xx/axes.scale_x }))/0.0000001;
          //coefficiente angolare della retta tangente nel punto i
          var m=derivative;
          //coefficiente q della retta tangente nel punto i
          var q=Parser.evaluate(func, { x: xx/axes.scale_x })- derivative*xx/axes.scale_x;

          ctx.beginPath();
          ctx.strokeStyle=rgb(r,10,100);
          //disegno la retta tangente
          ctx.moveTo(((axes.ymin-q)/m)*axes.scale_x-axes.xmin_px,HEIGHT);
          ctx.lineTo(((axes.ymax-q)/m)*axes.scale_x-axes.xmin_px,0);
          ctx.stroke();

          //valore della derivata moltiplicata per la scala (in pixel)
          var scaled_derivative = axes.scale_y * (Parser.evaluate(func, { x: xx/axes.scale_x+0.0000001 }) - Parser.evaluate(func, { x: xx/axes.scale_x }))/0.0000001;
          ctxDer.beginPath();
          ctxDer.fillStyle = rgb(r,10,100);
          //disegno la funzione derivata
          ctxDer.arc(xx-axes.xmin_px,HEIGHT-axes.ymin_px-scaled_derivative,2,0,2*Math.PI);
          ctxDer.fill();

```

- drawDifferenceQuotient()

```
        var functan="(x-("+xfis+"))/(("+xmob+")-("+xfis+"))*(("+ymob+")-("+yfis+"))+("+yfis+")"; //retta passante per 2 punti
        ctx.beginPath();
        var r=Math.round((px_counter)/(interval*axes.scale_x)*255);
        ctx.strokeStyle=rgb(r,10,100);
        ctx.lineWidth=2;
        ctx.moveTo(0,HEIGHT-axes.ymin_px-axes.scale_y*Parser.evaluate(functan, { x: axes.xmin }));
        ctx.lineTo(WIDTH,HEIGHT-axes.ymin_px-axes.scale_y*Parser.evaluate(functan, { x: axes.xmax }));
        ctx.stroke();
        ctx.beginPath();
        ctx.fillStyle="black";
        ctx.arc(xfis*axes.scale_x-axes.xmin*axes.scale_x,HEIGHT-axes.ymin_px-yfis*axes.scale_y,5,0,2*Math.PI);
        ctx.fill();
        ctx.closePath();
        ctx.beginPath();
        ctx.fillStyle="black";
        ctx.arc(xmob*axes.scale_x-axes.xmin*axes.scale_x,HEIGHT-axes.ymin_px-ymob*axes.scale_y,5,0,2*Math.PI);
        ctx.fill();
        ctx.closePath();
      }
```

- drawImproperIntegral

```
        yy=HEIGHT-axes.ymin_px-axes.scale_y*Parser.evaluate(func, { x: xx/axes.scale_x });
        ctx.fillStyle=rgb(0,128,255);
        ctx.beginPath();
        //disegno l'area sottesa alla funzione in base a come è impostata la velocità
        ctx.fillRect(xx+WIDTH/2,HEIGHT-axes.ymin_px,2,-axes.scale_y*Parser.evaluate(func, { x: xx/axes.scale_x }));
        ctx.fill();
        if(speed!="a=-b^2")
          ctx.fillRect(WIDTH/2-xx,HEIGHT-axes.ymin_px,2,-axes.scale_y*Parser.evaluate(func, { x: -xx/axes.scale_x }));
        else
          for (var i=0;i<=(Math.pow(((xx+1)/axes.scale_x),2)-Math.pow((xx/axes.scale_x),2))*axes.scale_x;i++){
             ctx.fillRect(WIDTH/2-Math.pow((xx/axes.scale_x),2)*axes.scale_x-i,HEIGHT-axes.ymin_px,2,-axes.scale_y*Parser.evaluate(func, { x: -Math.pow((xx/axes.scale_x),2)-i/axes.scale_x }))
             ctx.fill();
            }

        ctxDer.fillStyle = rgb(0,128,255);
        //disegno la funzione integrale
        ctxDer.beginPath();
        ctxDer.arc(WIDTH/2+px_counter,HEIGHT-axes.ymin_px-(integral.values[px_counter]*axes.scale_y)/integral.y_axes_correction,2,0,2*Math.PI);
        ctxDer.fill();
```

Per la rappresentazione e la valutazione della derivata prima non sono stati utilizzati tool di calcolo simbolico ma il valore f'(x0) è stato approssimato con un rapporto incrementale in avanti del primo ordine:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

con h = 10^{-7} . È noto che l'errore prodotto in questa approssimazione è proporzionale ad h. (si veda **[2]** per l'analisi dell'errore).

Anche per il calcolo della funzione integrale e dell'integrale improprio è stato utilizzato un metodo numerico, in particolare il metodo dei trapezi secondo cui:

$$I(f) = \int_a^b f(x)dx \approx \frac{f(b) + f(a)}{2} (b - a) = I_T$$

L'errore prodotto dalla formula dei trapezi è proporzionale a $(b-a)^3$[**2**]. Il valore della funzione integrale è calcolato sfruttando la forma composita dei trapezi, secondo cui l'intervallo di integrazione è suddiviso in sottointervalli I_k di ampiezza variabile h_k e l'errore finale prodotto sarà dato dalla somma degli errori generati su ogni singolo intervallo.

Infine in ***DrawUtilities.js*** sono contenute alcune funzioni utili per la creazione dei disegni e per la gestione degli errori.

Questi tre script vengono sfruttati dalle pagine html dove sono visualizzati i vari canvas e dove sono contenuti alcuni metodi che si occupano della configurazione della pagina, l'avvio e la manutenzione dell'animazione e la gestione degli input dall'utente.

Una parte del codice molto significativa è quella in cui viene richiamato il metodo Draw(), in quanto al variare dell'intervallo di recall (in ms) dettato dalla costante "TIME" variano alcuni parametri molto delicati dell'animazione, infatti viene direttamente modificato il tempo che intercorre tra un frame ed il successivo ed indirettamente il tempo necessario al completamento dell'intera animazione, è importante quindi scegliere un valore della costante "TIME" sufficientemente grande in modo tale che l'applicazione abbia il tempo necessario per ridisegnare ogni frame prima della chiamata successiva ma che allo stesso tempo sia sufficientemente piccolo da rendere l'animazione fluida e priva di blocchi.

```
1. // se entro nella funzione init dopo aver gia svolto almeno un animazione blocco il vecchio setInterval e rimuovo i listener in preparazione della nuova animazione
2.                 if (px_counter>dom.last_pixel-dom.first_pixel+10){
3.                   first_time=true;
4.                   clearInterval(callID);
5.                   document.getElementById("Circle").removeEventListener("mousemove", getPosition, false);
6.                   document.getElementById("Circle").removeEventListener("mousedown", getPosition, false);
7.                 }
8.                 //chiamo il nuovo setInterval e mi salvo l'id.
9.                 callID=setInterval(draw,TIME,axes,fx,dom);
```

L'istruzione alla riga 9 avvia la chiamata ogni 20 millisecondi del metodo draw() che si occupa appunto di ridisegnare il canvas, finché non viene eseguito il ClearInterval(callID) che interrompe il ciclo di chiamate.

In aggiunta al codice prodotto sono stati utilizzati alcuni strumenti esterni come "Bootstrap" **[8]**, un framework css che offre un insieme di elementi grafici, stilistici, di impaginazione e Javascript pronti all'uso e "Javascript Expression Evaluator" **[9]**, una libreria Javascript in grado di restituire il valore in un punto assunto da una funzione matematica inserita sotto forma di stringa.

4.2 Testing

La fase di testing consiste nel sottoporre il software ad una serie di prove atte a ad individuare eventuali errori, difetti o malfunzionamenti.

In questo caso il testing è stato applicato a 5 stadi intermedi dello sviluppo, corrispondenti alla generazione delle 5 animazioni.

Per ognuno dei 5 stadi sono state analizzate le risposte del sistema di fronte ad un input non corretto da parte dell'utente:

Input	**Risposta del sistema**
Inserimento di una stringa non valida nel campo "funzione"	Il sistema crea un pannello d'allarme che avverte l'utente che la funzione è sintatticamente scorretta. Proponendo la lettura della documentazione.
Inserimento di "x-min" maggiore di "x-max".	Il sistema crea un pannello d'allarme che avverte l'utente che il range degli assi non è corretto.
Inserimento di "y-min" maggiore di "y-max" il sistema	Il sistema crea un pannello d'allarme che avverte l'utente che il range degli assi non è corretto.
***solo per Integrale/integrale Improprio** Inserimento di una funzione illimitata.	Il sistema crea un pannello d'allarme che avverte l'utente che sono accettate solo funzioni limitate.
***solo per Integrale Improprio** Inserimento di un range per l'asse delle ascisse non simmetrico.	Il sistema crea un pannello d'allarme che avverte l'utente che sono accettati solo range dell'asse delle ascisse simmetrici.

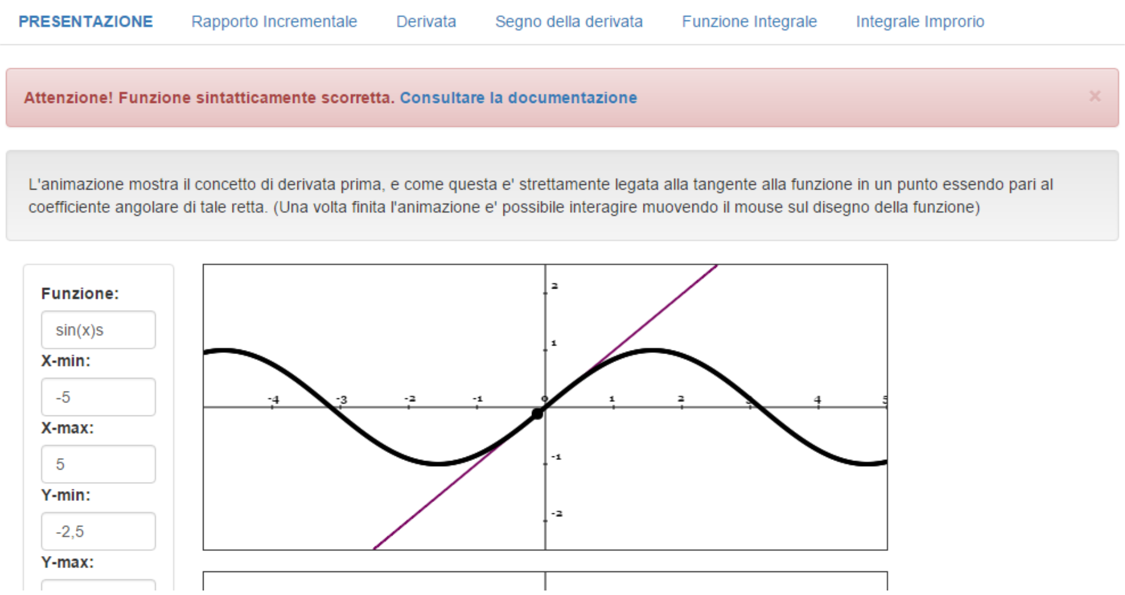
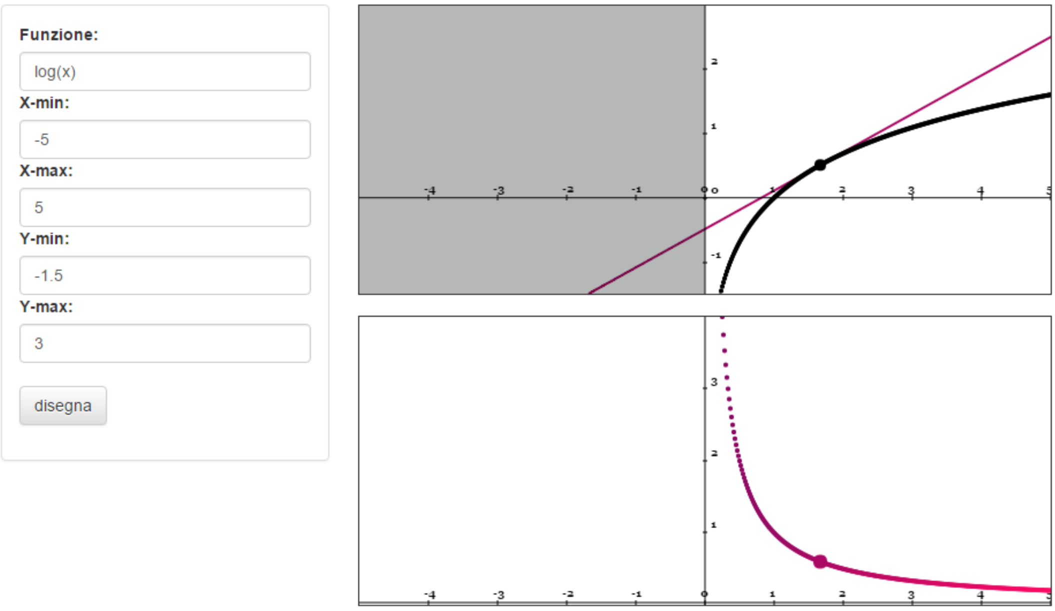


Figura 2 - Esempio Pannello d'allarme.

Successivamente, per ogni stadio, sono state effettuate diverse prove confrontando i risultati ottenuti con quelli attesi.

Fra le prove effettuate, sono stati analizzati anche casi particolari come funzioni non definite sul tutto il range delle x o funzioni illimitate:

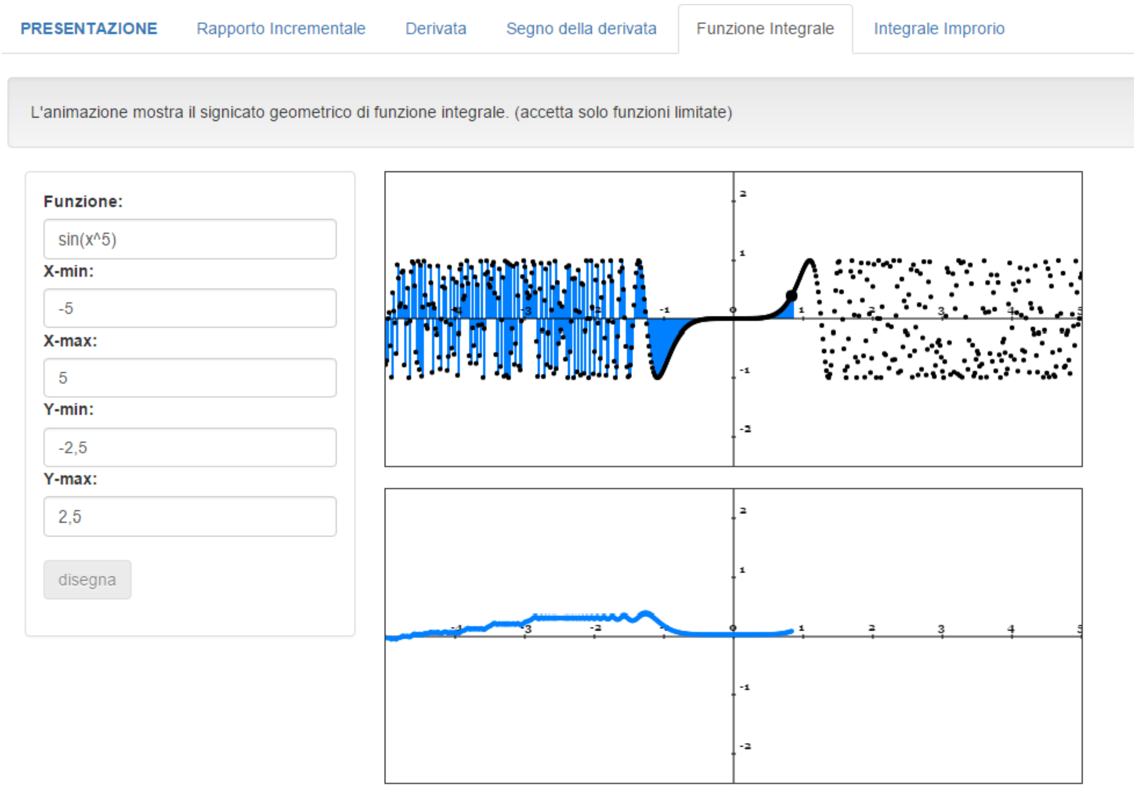
Es. Log(x)



I test effettuati hanno evidenziato anche alcuni limiti dell'applicazione:

Un primo limite riguarda le funzioni con un forte gradiente o fortemente oscillanti, esse infatti non sono disegnabili correttamente nei canvas per via del numero limitato di pixel di cui si è a disposizione.

Es.sin(x^5)



Un altro limite rilevato durante i test riguarda il modo in cui viene generato l'asse delle ordinate del secondo canvas, infatti per far sì che la funzione derivata prima e la funzione integrale siano contenute e ben visibili nel grafico, si è provveduto a modificare dinamicamente l'asse delle y, questa procedura funziona correttamente per funzioni limitate, ma presenta delle imprecisioni di fronte a funzioni tendenti ad infinito.

Per risolvere questo problema è stata aggiunta una *checkbox* in grado di disattivare la modifica dinamica dell'asse delle ordinate e di proporre all'utente l'inserimento manuale di esso.

4.3 Performance

Una fase molto importante dello sviluppo di questo progetto è stata lo studio delle performance grafiche.

Una prima analisi ha evidenziato come su pc leggermente datati, in particolare su quelli dotati di sistemi operativi basati su linux, o ancor di più su smartphone, il tempo di 20 millisecondi che intercorreva tra un frame ed il successivo non era sempre sufficiente per effettuare tutti i calcoli necessari portando così alla perdita di alcuni frame e di conseguenza ad un'animazione poco fluida.

Non potendo intervenire sul tempo di recall, in quanto un aumento di esso avrebbe portato ad un framerate troppo basso, si è deciso di intervenire sui calcoli effettuati per ridisegnare ogni frame, cercando di ridurli, per far questo sono stati applicati due processi:

1. Suddivisione dei canvas in più livelli.
2. Riduzione degli oggetti da disegnare.

Il primo processo è stato effettuato scomponendo il canvas principale in più canvas sovrapposti, uno statico, ridisegnato una sola volta prima dell'inizio dell'animazione e contenente gli assi e la funzione inserita dall'utente, uno dinamico ridisegnato ad ogni chiamata del metodo draw() e contenente l'animazione vera e propria o una parte di essa, ed eventualmente un terzo, contenente una parte dell'animazione non completamente ridisegnata ma solo modificata.

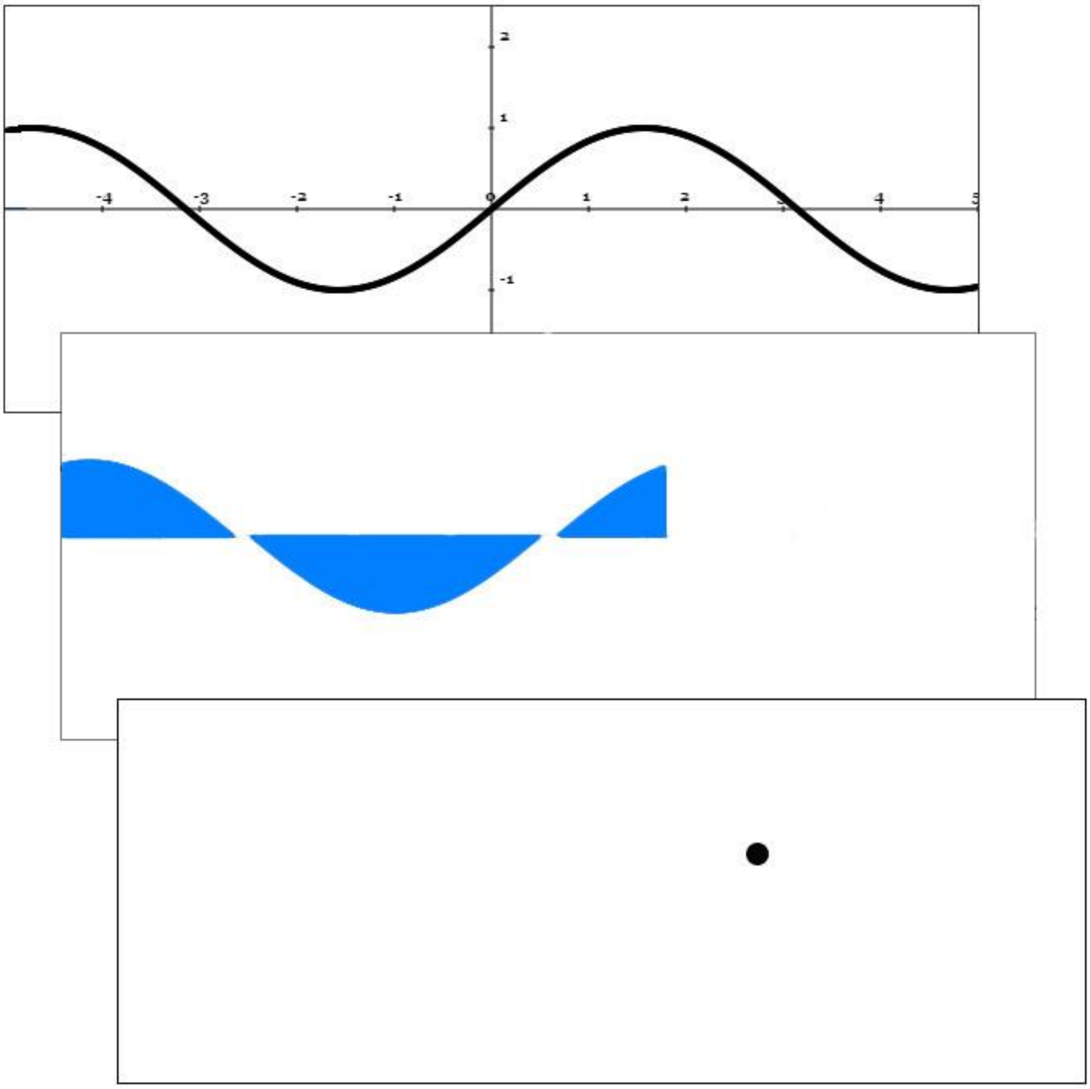


Figura 3 - Suddivisione canvas in livelli

Il primo processo ha portato ad un sostanziale miglioramento delle prestazioni ed è stato sufficiente per rendere fluide le animazioni di funzione derivata prima, rapporto incrementale e segno della derivata.

Le animazioni riguardanti invece la funzione integrale e l’integrale improprio hanno necessitato di ulteriore processo di miglioramento:

è stato rivisto il modo di disegnare l’area sottesa alla funzione, in un primo approccio ad ogni chiamata del metodo draw, il canvas veniva cancellato completamente e veniva ridisegnato un rettangolo per ogni pixel dall’inizio della funzione al punto in cui era arrivata l'animazione portando quindi a dover disegnare fino a 600 oggetti in una sola chiamata, l'approccio migliorativo è stato invece quello di non cancellare il canvas e di disegnare ad ogni chiamata solo il rettangolo corrispondente al punto in cui l'animazione era arrivata. Questo ha portato un ulteriore miglioramento delle prestazioni dell'animazione di integrale e integrale improprio rendendo anch'esse sufficientemente fluide.

4.4 Refactoring

L’ultima fase di sviluppo del sistema è stata la fase di refactoring, un processo di modifica del codice tale da non modificare il comportamento del software ma di migliorarne la struttura interna, rendendone più semplice la comprensione e di conseguenza semplificando eventuali modifiche future.

Per il refactoring sono state utilizzate principalmente le tecniche di *renaming* e di *raplace variable with query*.

La tecnica di *renaming* consiste nel rinominare tutte le entità i cui nomi non rivelano chiaramente lo scopo per cui sono state create.

La tecnica di *replace variable with query* consiste invece nel sostituire le variabili temporanee utilizzate nel codice con l’espressione o il valore ad esse associato.

4.5 Funzionalità aggiunte

Dopo il completamento dell’applicazione si è deciso di implementare una nuova funzionalità, nei casi di rapporto incrementale, derivata prima e segno della derivata si è aggiunta infatti la possibilità per l’utente di interagire con le animazioni muovendosi con il mouse all’interno del canvas. Per far tutto ciò sono stati aggiunti dei *listener* al canvas in grado di rilevare il movimento e la pressione del mouse ed è stata creata una nuova funzione chiamata *getPosition()* utile a calcolare le coordinate del mouse relativamente al canvas principale e di inviarle ad un'altra funzione che si occupa di ridisegnare il frame dell’animazione corrispondente alla coordinata x del mouse.

I listener sono aggiunti nel momento in cui l’animazione si conclude e vengono rimossi nel momento in cui riparte una nuova animazione.

```
document.getElementById("BackgroundFunction").addEventListener("mousemove", getPosition, false);
document.getElementById("BackgroundFunction").addEventListener("mousedown", getPosition, false);
```

```
document.getElementById("BackgroundFunction").removeEventListener("mousemove", getPosition, false);
document.getElementById("BackgroundFunction").removeEventListener("mousedown", getPosition, false);
```

```
function getPosition(event)
    {
    if(!$("#start").prop("disabled")){
    var x = new Number();
    var y = new Number();
    var canvas = document.getElementById("BackgroundFunction");
    var menu=document.getElementById("menu");
    var col=document.getElementById("col1")
    if (event.x != undefined && event.y != undefined)
    {
      x = event.pageX;
      y = event.pageY;
    }
    else
    {
      x = event.clientX + document.body.scrollLeft +
          document.documentElement.scrollLeft;
      y = event.clientY + document.body.scrollTop +
          document.documentElement.scrollTop;
    }

    if (col.offsetWidth+canvas.offsetWidth<$(window).width())
        x-= canvas.offsetLeft+col.offsetWidth+col.offsetLeft;
    else
        x-= canvas.offsetLeft;
    y-= canvas.offsetTop+menu.offsetHeight+menu.offsetTop;


    mouseX=x;
    mouseY=y;
    drawInteractive(mouseX,mouseY);
    }
    }
```

Questa nuova funzionalità è stata realizzata in modo che fosse usufruibile anche da smartphone, è per questo che oltre al *listener* del movimento è stato aggiunto anche quello che rileva la pressione del mouse, infatti per un dispositivo mobile questo evento si scatena premendo sullo schermo in corrispondenza del canvas.

4.6 Esempi

$$\sin(\mathrm{x}) + \frac{x^2}{10}$$

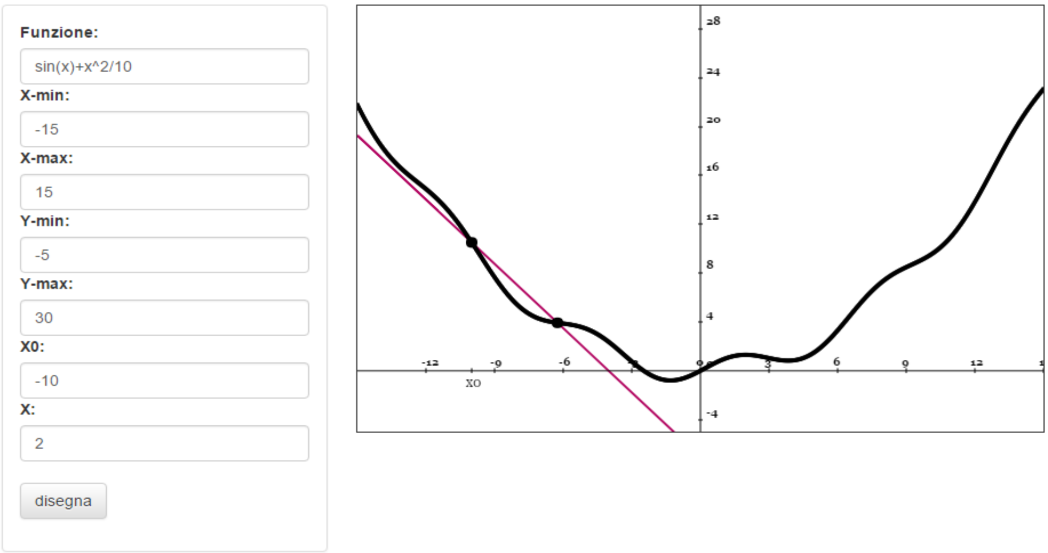


Figura 4 - Rapporto Incrementale

$$cos(x) + \frac{x}{10}$$

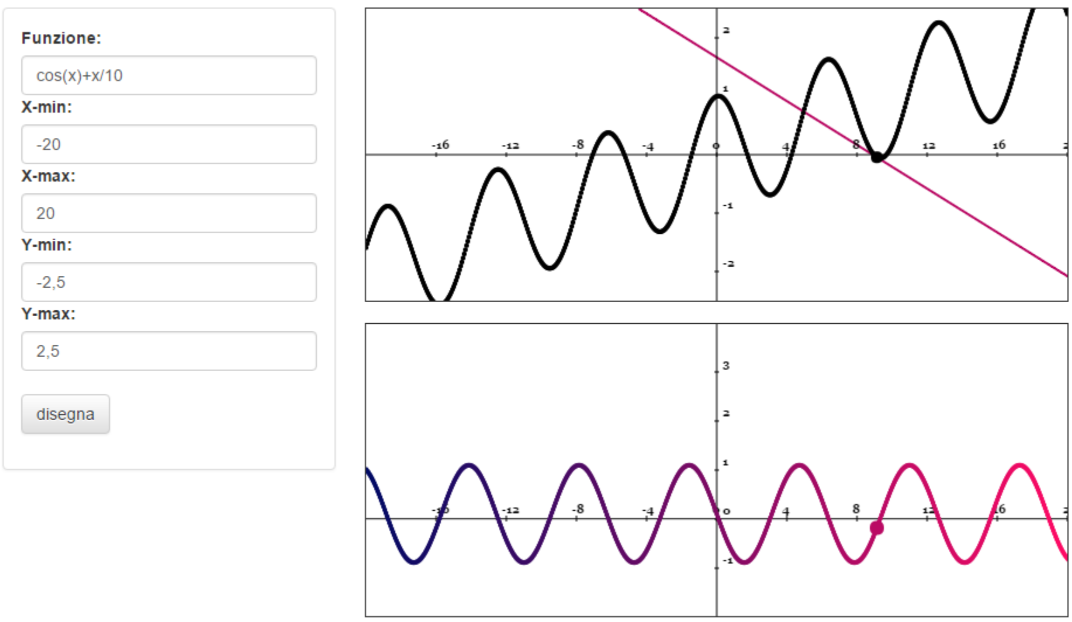


Figura 5 - Funzione derivata prima

$$sin(3x)$$

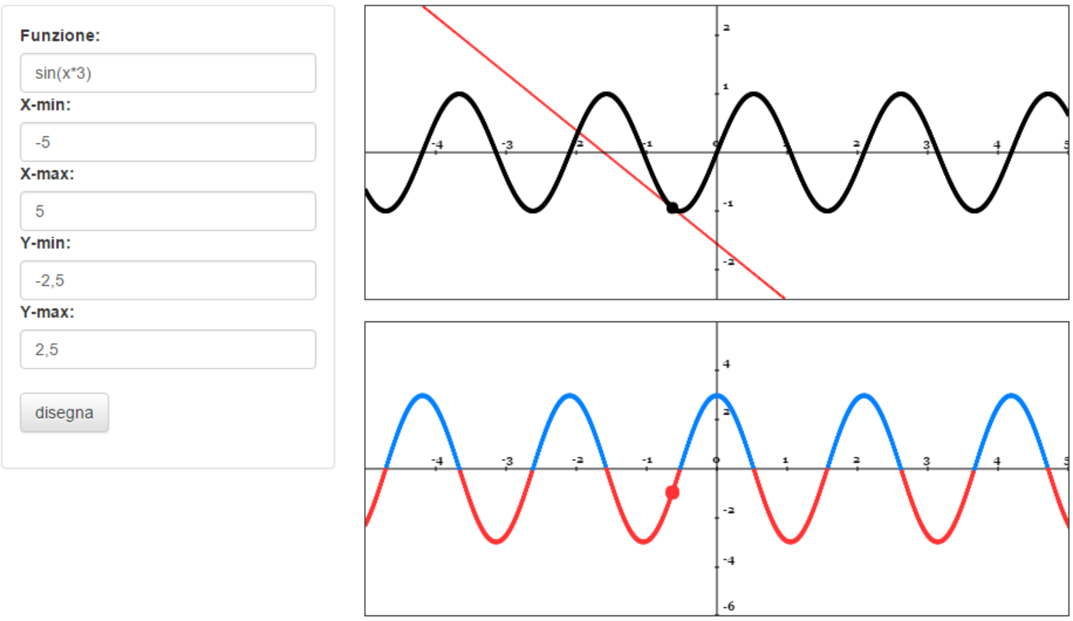


Figura 6 - Segno della derivata

$$\frac{sin(x)}{x}$$

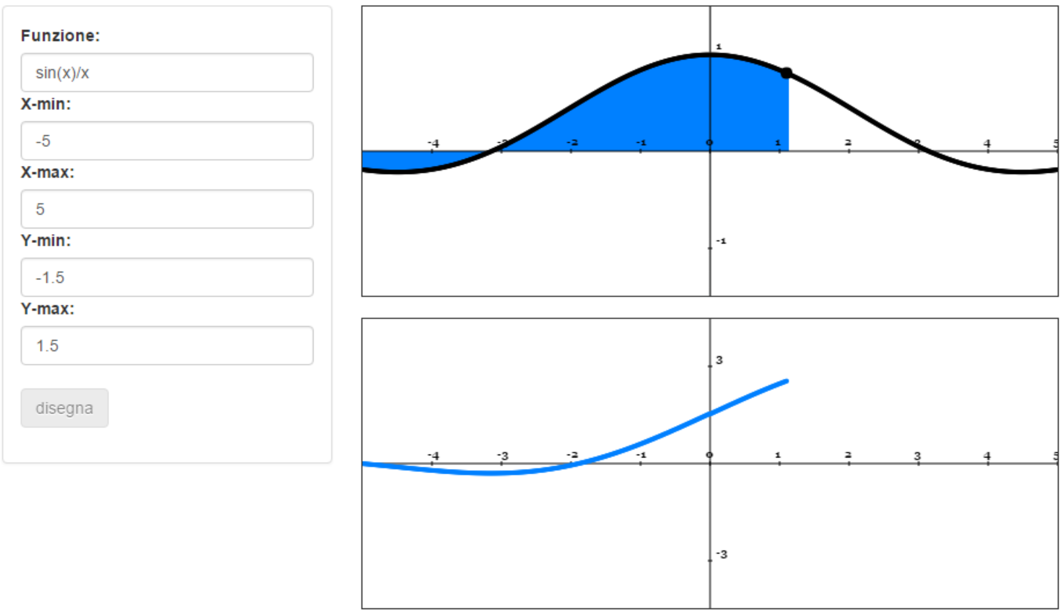


Figura 7 - Funzione Integrale

$$\frac{2x}{1+x^2}$$

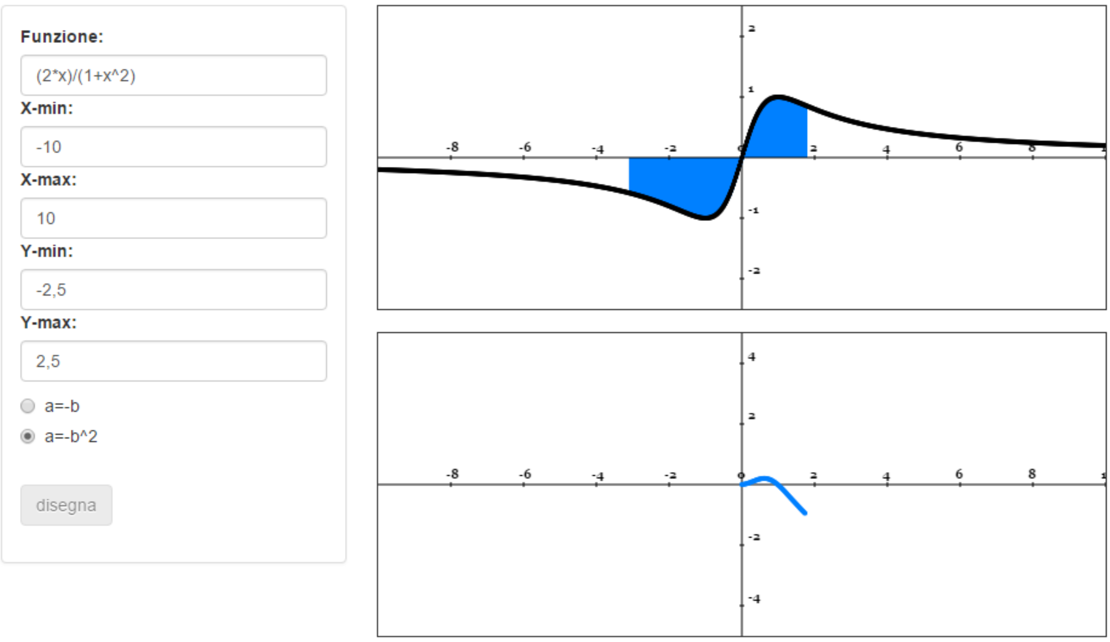


Figura 8 - Integrale Improprio

Capitolo 5:
CONCLUSIONI

Nella presente tesi ho sviluppato un software per la generazione dinamica di 5 animazioni con HTML5 Canvas per la rappresentazione grafica della derivata prima in un punto, della funzione derivata prima, del segno della derivata prima (in relazione a crescenza/decrescenza della funzione), della funzione integrale e dell'integrale improprio sulla retta reale.

L'implementazione ha seguito varie fasi.

Anzitutto ho svolto l'analisi dei requisiti e dei casi d'uso in cui ho chiarito i punti chiave del funzionamento dell'applicazione e ho preso alcune scelte (fra le quali quella di utilizzare i Canvas) che hanno garantito un'elevata portabilità, l'applicazione è infatti utilizzabile con dispositivi mobili.

Quindi sono passato alla stesura del codice, al testing ed al refactoring, in cui il codice ho elaborato e testato il codice, facendo particolare attenzione anche alla correttezza sintattica dei dati inseriti.

Infine ho rimaneggiato il codice per migliorarne la leggibilità.

Per la stesura del codice sono stati utilizzati alcuni strumenti, quali *Sublime Text 3* **[5]** come editor di testo, il framework css *Bootstrap* **[8]** e la libreria *Javascript Expression Evaluator* **[9]** che ha permesso il parsing delle funzioni matematiche.

È possibile utilizzare l'applicazione alla pagina web http://claudiocatterina.altervista.org/MathCanvas

BIBLIOGRAFIA E SITOGRAFIA

1. C. Canuto, A. Tabacco, Analisi Matematica 1, Ed. Springer Italia, 4a edizione, 2014
2. A. Quarteroni, F. Saleri, P. Gervasio, Calcolo Scientifico. Springer Italia, 5a edizione, 2012)
3. http://www.html.it/pag/19262/introduzione77/
4. https://msdn.microsoft.com/it-it/library/gg193983(v=vs.85).aspx
5. http://www.sublimetext.com/3
6. https://github.com/Warin/Sublime/tree/master/DocBlockr
7. https://www.npmjs.com/package/doxx
8. http://getbootstrap.com/
9. http://silentmatt.com/javascript-expression-evaluator/